To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Cautions

User's Manual

H8/3802 Series E6000 Emulator HS3800EPI60H

User's Manual

Renesas Microcomputer Development Environment System

Renesas Electronics
www.renesas.com

Rev.1.0 2000.09

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

IMPORTANT INFORMATION

READ FIRST

• READ this user's manual before using this E6000 emulator.

• <u>KEEP the user's manual handy for future reference.</u>

Do not attempt to use the E6000 emulator until you fully understand its mechanism.

E6000 emulator:

Throughout this document, the term "E6000 emulator" shall be defined as the E6000 emulator, user system interface cable, PC interface board, and optional boards produced only by Hitachi, Ltd. excluding all subsidiary products.

The user system or a host computer is not included in this definition.

Purpose of the E6000 emulator:

This E6000 emulator is a software and hardware development tool for systems employing the Hitachi microcomputer H8/3802 series (hereafter referred to as MCU). This E6000 emulator must only be used for the above purpose.

Improvement Policy:

Hitachi, Ltd. (including its subsidiaries, hereafter collectively referred to as Hitachi) pursues a policy of continuing improvement in design, functions, performance, and safety of the E6000 emulator. Hitachi reserves the right to change, wholly or partially, the specifications, design, user's manual, and other documentation at any time without notice.

Target User of the E6000 emulator:

This E6000 emulator should only be used by those who have carefully read and thoroughly understood the information and restrictions contained in the user's manual. Do not attempt to use the E6000 emulator until you fully understand its mechanism.

It is highly recommended that first-time users be instructed by users that are well versed in the operation of the E6000 emulator.

LIMITED WARRANTY

Hitachi warrants its E6000 emulators to be manufactured in accordance with published specifications and free from defects in material and/or workmanship. Hitachi, at its option, will repair or replace any E6000 emulators returned intact to the factory, transportation charges prepaid, which Hitachi, upon inspection, determine to be defective in material and/or workmanship. The foregoing shall constitute the sole remedy for any breach of Hitachi's warranty. See the Hitachi warranty booklet for details on the warranty period. This warranty extends only to you, the original Purchaser. It is not transferable to anyone who subsequently purchases the emulator product from you. Hitachi is not liable for any claim made by a third party or made by you for a third party.

DISCLAIMER

HITACHI MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, ORAL OR WRITTEN, EXCEPT AS PROVIDED HEREIN, INCLUDING WITHOUT LIMITATION THEREOF, WARRANTIES AS TO MARKETABILITY, MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR USE, OR AGAINST INFRINGEMENT OF ANY PATENT. IN NO EVENT SHALL HITACHI BE LIABLE FOR ANY DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE, OR LOSSES OR EXPENSES RESULTING FROM ANY DEFECTIVE E6000 EMULATOR, THE USE OF ANY E6000 EMULATOR, OR ITS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCEPT AS EXPRESSLY STATED OTHERWISE IN THIS WARRANTY, THIS E6000 EMULATOR IS SOLD "AS IS ", AND YOU MUST ASSUME ALL RISK FOR THE USE AND RESULTS OBTAINED FROM THE E6000 EMULATOR.

State Law:

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may have other rights which may vary from state to state.

The Warranty is Void in the Following Cases:

Hitachi shall have no liability or legal responsibility for any problems caused by misuse, abuse, misapplication, neglect, improper handling, installation, repair or modifications of the E6000 emulator without Hitachi's prior written consent or any problems caused by the user system.

All Rights Reserved:

Other Important Things to Keep in Mind:

1. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.

2. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi.

Figures:

Some figures in this user's manual may show items different from your actual system.

Limited Anticipation of Danger:

Hitachi cannot anticipate every possible circumstance that might involve a potential hazard. The warnings in this user's manual and on the E6000 emulator are therefore not all inclusive. Therefore, you must use the E6000 emulator safely at your own risk.

SAFETY PAGE

READ FIRST

- **READ this user's manual before using this E6000 emulator.**
- **<u>KEEP the user's manual handy for future reference.</u>**

Do not attempt to use the E6000 emulator until you fully understand its mechanism.

DEFINITION OF SIGNAL WORDS

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

CAUTION

CAUTION used without the safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in property damage.

NOTE emphasizes essential information.

WARNING

Observe the precautions listed below. Failure to do so will result in a FIRE HAZARD and will damage the user system and the E6000 emulator or will result in PERSONAL INJURY. The USER PROGRAM will be LOST.

1. **Do not repair or remodel the emulator product by yourself for electric shock prevention and quality assurance.**
2. **Always switch OFF the E6000 emulator and user system before connecting or disconnecting any CABLES or PARTS.**
3. **Always before connecting any CABLES, make sure that pin 1 on both sides are correctly aligned.**
4. **Supply power according to the power specifications and do not apply an incorrect power voltage. Use only the provided power cable.**

CAUTION

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

About This Manual

This emulator (HS3800EPI60H) supports the following MCUs. In this manual, only the MCU names are shown.

Devices to be Supported	MCU
H8/3800, 3801, and 3802	H8/3802 series

This manual explains how to set up and use the E6000 emulator for the H8/3802 series of microcomputers. This manual describes the debugging platform.

Section 1, *Introduction*, gives a rapid introduction to the system's facilities, including an overview of the main emulation features provided by the E6000 emulator and the Hitachi debugging interface (HDI) software that provides access to them.

Section 3, *Hardware*, explains how to connect the E6000 emulator to an external user system.

Section 4, *Tutorial*, then introduces each of the E6000 emulator's main features by showing how to load and debug a simple C program. The tutorial program is supplied on disk so that you can follow the steps on your own system to learn first-hand how it operates.

Section 5, *Reference*, gives detailed information about the features of the HDI software.

Assumptions

This manual assumes that you already have a working knowledge of the following:

- The procedures for running and using MS-DOS® and Windows® programs.

Related Manuals

- Hitachi Debugging Interface User's Manual
- User System Interface Cable User's Manual
- PC Interface Board User's Manual
  - ISA Bus Interface Board User's Manual (HS6000EII01HE)
  - PCI Bus Interface Board User's Manual (HS6000EIC01HE or HS6000EIC02HE)
  - PCMCIA Interface Card User's Manual (HS6000EIP01HE)
  - Description Notes on Using LAN Adapter for E6000/E8000 Emulator (HS6000ELN01HE)

Conventions

This manual uses the following typographical conventions:

Style	Used for
`computer`	Text that you type in.
parameter	A label representing the actual value you should type as part of a command.
bold	Names of menus, menu commands, buttons, dialog boxes, text that appears on the screen, and windows that appear on the screen.

Trademarks

- Microsoft®, MS-DOS®, Windows®, and Windows NT® are registered trademarks of Microsoft Corporation in the United States and other countries.
- IBM is a registered trademark of International Business Machines Corporation.
- This manual describes the operating environment as Microsoft® Windows® 98 English version on the IBM PC.

Contents

RENESAS

RENESAS

Figures

Tables

Section 1 Introduction

The E6000 emulator is an advanced realtime in-circuit emulator which allows programs to be developed and debugged for the H8/3802 series of microcomputers.

The E6000 emulator can either be used in stand-alone mode, for software development and debugging, or connected via a user system interface cable to a user system, for debugging user hardware.

The E6000 emulator works with the Hitachi debugging interface (HDI), a Windows®-based interface program. This provides a powerful range of commands for controlling and interrogating the emulator hardware, with a choice of either fully interactive or automated debugging.

1.1 Debugging Features

1.1.1 Breakpoints

The E6000 emulator provides a comprehensive range of alternative types of breakpoints, to give you the maximum flexibility in debugging applications and user system hardware.

Hardware Breakpoints: Up to 12 breakpoints can be defined using the event and range channels in the complex event system (CES). For more information about the hardware breakpoints see section 1.2, Complex Event System (CES).

Program Breakpoints (PC Breakpoints): Up to 256 program breakpoints can be defined. These program breakpoints are set by replacing the user instruction by a BREAK instruction.

1.1.2 Trace

The E6000 emulator incorporates a powerful realtime trace facility which allows you to examine MCU activity in detail. The realtime trace buffer holds up to 32768 bus cycles, and it is continuously updated during execution. The buffer is configured as a rolling buffer, which can be stopped during execution and read back by the host computer without halting emulation.

The data stored in the trace buffer is displayed in both source program and assembly languages for ease of debugging. However, if trace filtering is used then only assembly language can be displayed.

The buffer can be set up to store all bus cycles or just selected cycles. This is called trace acquisition and uses the complex event system (CES) to select the parts of the program you are interested in; see section 1.2, Complex Event System, for more information.

It is also possible to store all bus cycles and then just look at selected cycles. This is called trace filtering.

1.1.3 Execution Time Measurements

The E6000 emulator allows you to make measurements of the total execution time, or to measure the time of execution between specified events in the complex event system. You can set the resolution of the timer to any of the following values:

20 ns, 125 ns, 250 ns, 500 ns, 1 µs, 2 µs, 4 µs, 8 µs or 16 µs.

At 20 ns the maximum time that can be measured is six hours, and at 16µs the maximum time is about 200 days.

1.2 Complex Event System (CES)

In most practical debugging applications the program or hardware errors that you are trying to debug often only occur under a certain very restricted set of circumstances. For example, a hardware error may only occur after a specific area of memory has been accessed. Tracking down such problems using simple PC breakpoints can be very time consuming.

The E6000 emulator provides a very sophisticated system for giving a precise description of the conditions you want to examine, called the complex event system. This allows you to define events which depend on the state of a specified combination of the MCU signals.

The complex event system provides a unified way of controlling the trace, break, and timing functions of the E6000 emulator.

1.2.1 Event Channels

The event channels allow you to detect when a specified event has occurred. The event can be defined as a combination of one or more of the following:

- Address or address range.
- Outside the address range
- Data, with an optional mask.
- Read or write.
- MCU access type (instruction prefetch, data fetch, etc).
- MCU access area (internal ROM, internal RAM, etc).
- A signal state on one or more of the four external probes.

- You can specify that the event must be triggered a certain number of times before the event is activated.
- Delay cycles after an event.

Up to eight events can be combined into a sequence, in which each event is either activated or deactivated by the occurrence of the previous event in the sequence. For example, you can cause a break if an I/O register is written to after a specified area of RAM has been accessed.

1.2.2 Range Channels

The range channels provide a more limited set of options, and can be set up to be triggered on a combination of one or more of the following:

- Address or address range.
- Data, with an optional mask.
- Read or write.
- MCU access type (instruction prefetch, data fetch, etc).
- MCU access area (internal ROM, internal RAM, etc).
- A signal state on one or more of the four external probes.
- Delay cycles after an event.

1.2.3 Breaks and Timing

The complex event system can be used to control the following functions of the E6000 emulator:

Breaks: You use breaks to interrupt program execution when a specified event, or sequence of events, is activated. For example, you can set up a break to halt execution when the program is read from one address, and then written to another address. The break can also optionally be delayed by up to 65535 bus cycles.

Timing: You can perform precise timing measurements on sections of your program by setting up two events, and then timing the execution of the program between the activation of the first event and the activation of the second event.

1.3 Hardware Features

1.3.1 Memory

The E6000 emulator provides internal ROM/RAM memory as standard emulation memory.

The emulation memory can be mapped in one-byte units to the MCU address space. Each of memory can be specified using the **Configure Map…** command as user (Target) or emulator (internal ROM/RAM memory) and, in each case, the access can be specified as read-write, read-only, or guarded.

The definition of each type of memory is as follows:

Table 1.1 Memory Type

Memory type	Description
Internal	Uses the MCU internal memory.
Emulator	Uses the emulation board memory.

The contents of a specified block of memory can be displayed using the **Memory…** command. The contents of memory can be modified at any time, even during program execution and the results are immediately reflected in all other appropriate windows.

1.3.2 Operating Voltage and Frequency Specifications

Table 1.2 shows examples of the MCU operating voltage and frequency specifications supported by the E6000 emulator. Note that some MCUs do not guarantee the low-voltage operation or high-frequency operation.

Table 1.2 Operating Voltage and Frequency Specifications

MCU Type	Operating Voltage (V)	Operating Frequency Range (fosc) (MHz)
H8/3802 series	1.8-5.5	1.0-4.0
	2.7-5.5	1.0-10.0
	4.5-5.5	1.0-16.0

1.3.3 Clocks

The system clock and subclock can be programmed to the following frequencies.

Table 1.3 Clock Frequencies

Emulator	Emulation Clock	MCU	Frequency Selection
HS3800EPI60H	System clock	H8/3802 series	8 MHz, 2 MHz, 0.5 MHz, or target clock/2
	Subclock	H8/3802 series	32.768 kHz, 38.4 kHz, 307.2 kHz, or target subclock

1.3.4 External Probes

Up to four external probes can be connected to the E6000 emulator, to make use of signals from other parts of your user system hardware, and can be used to trigger the complex event system depending on whether the probe signal is low or high. When the external probes are not connected, the signals are fixed high. The state of a signal can be displayed in the trace window (high = 1, low = 0).

1.3.5 Environment Conditions

Observe the conditions listed in the following.

Table 1.4 Environment Conditions

Item	Specifications	
Temperature	Operating : +10 to +35°C	
	Storage : –10 to 50°C	
Humidity	Operating : 35 to 80% RH no condensation	
	Storage : 35 to 80% RH no condensation	
Ambient gases	Must be no corrosive gases	
AC input voltage	100 V to 240 V ± 10%	
AC input frequency	50/60 Hz	
AC current	0.6 A max.	
AC input cable*	HS3800EPI60H	HS3800EPI60HB
	100 V – 120 V (UL)	200 V – 240 V (BS)
User system voltage	1.8 V to 5.5 V	

Note: HS3800EPI60H must be used at AC100 V – 120 V input voltage.
HS3800EPI60HB must be used at AC200 V – 240 V input voltage.

1.3.6 Emulator External Dimensions and Mass

Dimensions: 219 × 160 × 54 mm

Mass: 970 g

Section 2 Setting Up

This section describes how to set up the E6000 emulator using the PC interface board and prepare it for use in conjunction with the Hitachi Debugging Interface (HDI).

This section explains how to:

- Set up the PC interface board.
- Set up the E6000 emulator.
- Install the HDI software and use it to check correct operation of the entire system.

2.1 Package Contents

The E6000 emulator is supplied in a package containing the following components.

- E6000 emulator.
- 5V 5A E6000 emulator power supply with AC cable.
- HDI installation disk (HS3800EPI60SF).
- Test program disk (HS3800EVI60SF).
- External probes.
- H8/3802 Series E6000 Emulator User's Manual (this manual).
- Hitachi Debugging Interface User's Manual.

Before proceeding you should check that you have all the items listed above, and contact your supplier if any are missing.

You also need an IBM PC or compatible as the host computer running Microsoft® Windows® 98 (hereinafter referred to as Windows® 98) or Windows NT® operating system, which is not supplied as part of the E6000 emulator package.

2.2 Installing the PC Interface Board

The PC interface board (HS6000EII01H) is a memory mapped board, and before using it you first need to reserve a block of memory addresses for use by the board. This ensures that other programs do not inadvertently use the PC interface hardware.

The allocated memory area must not overlap memory already allocated to other board. If attempted, the PC interface board and the E6000 emulator product will not operate correctly. At shipment, the memory area of PC interface board is allocated to the address range from H'D0000 to H'D3FFF.

2.2.1 Setting Up

- Start up the Windows® 98.
- Click the **My Computer** icon with the right button of the mouse to select **Properties** from the pop-up menu.

Then, the **System Properties** dialog box is displayed.

- Double-click the **Computer** icon on the **Device Manager** panel to open the **Computer Properties** dialog box.
- Click **Memory** on the **View Resource** panel to display the memory resource.

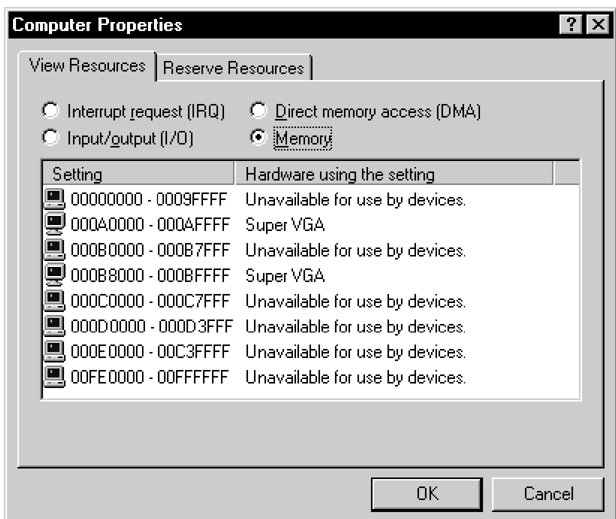


Figure 2.1 Computer Properties Dialog Box (Before Setting Up)

Memory areas not displayed on this dialog box can be used for the PC interface board. Table 2.1 shows the memory areas that can be specified by the switch of the PC interface board rear panel. In this table, select the areas that are not displayed on the **Computer Properties** dialog box. For example, if the area H'D8000 to H'DBFFF is selected, the corresponding switch number is 6.

Table 2.1 shows the memory switch and the address map of the PC interface board:

Table 2.1 Memory Switch and Address Map of PC Interface Board

Address Range	Switch
From H'C0000 to H'C3FFF	0
From H'C4000 to H'C7FFF	1
From H'C8000 to H'CBFFF	2
From H'CC000 to H'CFFFF	3
From H'D0000 to H'D3FFF (at shipment)	4
From H'D4000 to H'D7FFF	5
From H'D8000 to H'DBFFF	6
From H'DC000 to H'DFFFF	7
From H'E0000 to H'E3FFF	8
From H'E4000 to H'E7FFF	9
From H'E8000 to H'EBFFF	A
From H'EC000 to H'EFFFF	B

Take the following procedure so that the selected memory area is not used by Windows® 98.

- Click **Memory** and **Add** on the **Reserve Resource** panel.

Then, the **Edit Resource Setting** dialog box is displayed.

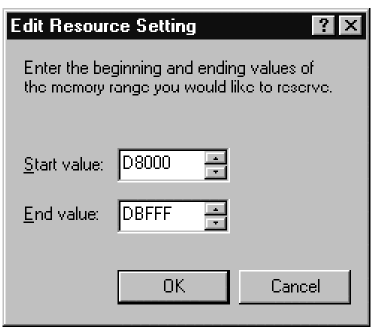


Figure 2.2 Edit Resource Setting Dialog Box

- Input the start and the end addresses of the selected memory area to **Start value** and **End value**.
- Turn off the host computer without restarting.
- Using a small screwdriver, rotate the selector switch in the back of the PC interface board so that the arrow points to the number corresponding to the range of addresses you have selected.
- Remove the cover from the host computer and install the PC interface board in a spare ISA slot.
- Put on the host computer cover.
- Connect the PC interface cable between the PC interface board and the PC interface connector on the E6000 emulator. Press each plug firmly until it clicks into position.
- Switch on the host computer.
- Confirm that the **System Reserved** is displayed for the memory area selected in the **Computer Properties** dialog box.

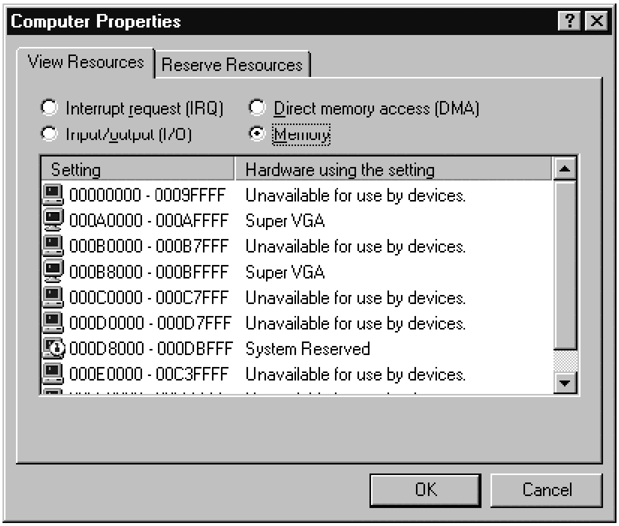


Figure 2.3 Computer Properties Dialog Box (After Setting up)

2.3 Setting Up the PC Interface Board on Windows NT® 4.0

The PC interface board uses the ISA bus slot, and therefore the host computer must have a spare ISA bus slot.

This section describes the general procedure for installing the PC interface board in the host computer. For details, refer to the manual of your host computer.

Starting Windows NT®:

- Execute **Start/Programs/Administrative Tools (Common)/Windows NT Diagnostics**.
- Click the **Memory** button in the **Resource** tab and, in the following form, make a note of the upper memory areas that have already been used.

#	Start	End	#	Start	End	#	Start	End
0			4			8		
1			5			9		
2			6			A		
3			7			B		

- Shut down Windows NT®.

Starting the Host Computer in Setup Mode:

For details on the setup mode, refer to the manual of your host computer.

- Check the upper memory areas that have already been used.

#	Start	End	#	Start	End	#	Start	End
0			4			8		
1			5			9		
2			6			A		
3			7			B		

The memory areas being used should be the same as those checked for Windows NT® above.

- Define the memory area for the PC interface board. Select one of the memory areas that correspond to the following PC interface board switch settings, and no other devices can access the selected memory area.

#	Start	End	#	Start	End	#	Start	End
0	H'C0000	H'C3FFF	4	H'D0000	H'D3FFF	8	H'E0000	H'E3FFF
1	H'C4000	H'C7FFF	5	H'D4000	H'D7FFF	9	H'E4000	H'E7FFF
2	H'C8000	H'CBFFF	6	H'D8000	H'DBFFF	A	H'E8000	H'EBFFF
3	H'CC000	H'CFFFF	7	H'DC000	H'DFFFF	B	H'EC000	H'EFFFF

If the **Intel P&P BIOS** disk is supplied with the host computer, define the memory area as follows:

- Start the host computer with the Intel P&P BIOS disk.
- Check the upper memory areas that have already been used, with **View/System Resources**.
- Add **Unlisted Card** with **Configure/Add Card/Others...**.
- Click **No** in the dialog box displayed because there is no .CFG file.
- Move to the **Memory [hex]** list box in the **Configure Unlisted Card** dialog box.
- Click the **Add Memory...** button to display the **Specify Memory** dialog box.
- Enter a memory area range that is not used by any other device and that corresponds to one of the PC interface board switch settings.

- Save the file.
- Exit the current setup program.
- Shut down the host computer (do not restart it) and turn off the power switch.
- Using a small screwdriver, rotate the switch in the rear panel of the PC interface board so that the arrow points to the number corresponding to the memory area you have selected.
- Remove the cover from the host computer and install the PC interface board in a spare ISA slot.
- Replace the host computer cover.
- Connect the PC interface cable between the PC interface board and the PC interface connector on the E6000 emulator. Press each plug firmly until it clicks into position.
- Switch on the host computer.

Starting Windows NT® in the Administrator Mode:

- Install the HDI Software as described in section 2.4, Installing the HDI Software.
- Execute **Start/Programs/Hdi/Setup ISA bus Board**.
  If the DOS prompt window does not open, open the DOS prompt window first, move to the directory where the HDI has been installed, then execute **SETUPISA.EXE**.

2.4 Installing the HDI Software

First install the HDI software from the installation disk as follows:

- Run Windows® if it is not already running.
- Close all other applications that are running.
- Insert HDI installation disk #1/3.
- Select **Run** from the **Start** menu.
- Type **`A:\Setup.exe`** and click **OK**:

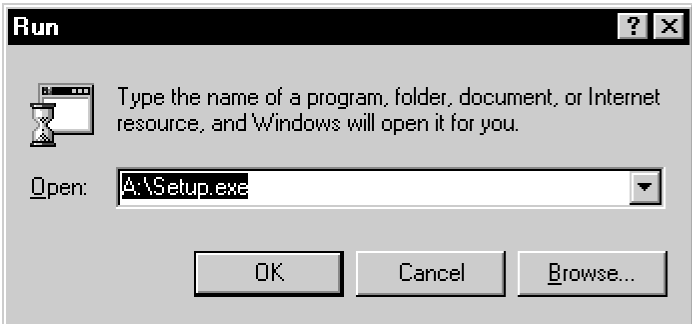


Figure 2.4 Selection Display for HDI Installation Disk

This runs the HDI installer, and the following **Welcome!** screen will be displayed:

Figure 2.5 HDI Installer Welcome! Screen

- Click **OK** to proceed with the installation.

The following dialog box then displays the **Read M**e file for the version of HDI you are installing:

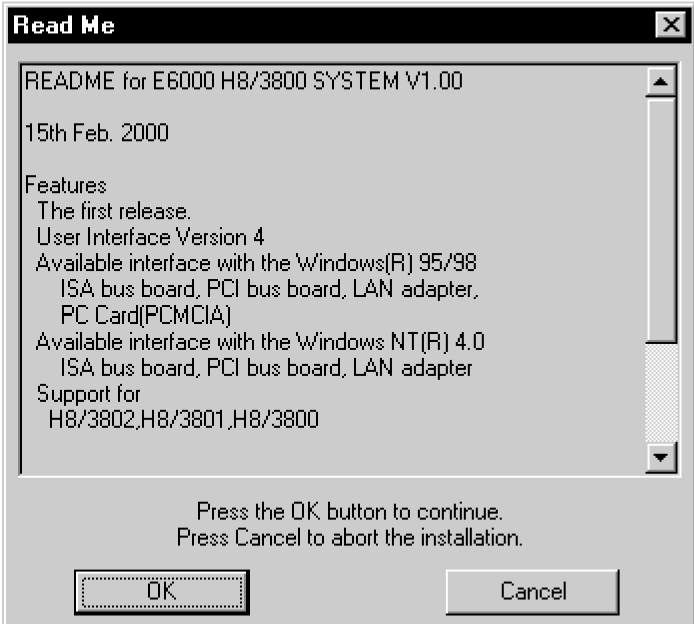


Figure 2.6 Read Me Dialog Box

- Check the **Read Me** file for any important information concerning the installation and then click **OK** to proceed.

The following dialog box then allows you to select a directory in which you can install HDI:

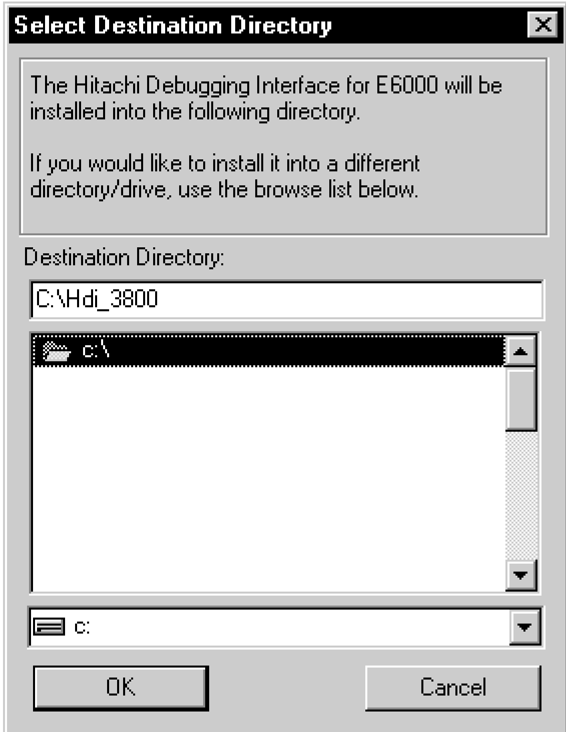


Figure 2.7 Select Destination Directory Dialog Box

- Click **OK** to install into the default directory, or specify an alternative directory and click **OK**.

The following dialog box then asks you whether backups should be made of files replaced by the installation:

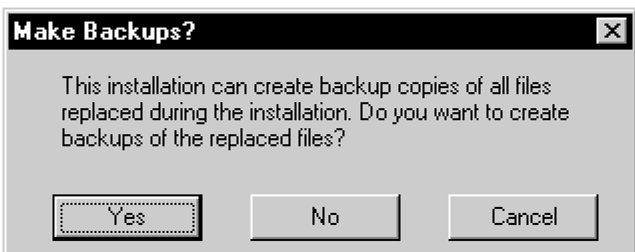


Figure 2.8 Make Backups? Dialog Box

- Click **Yes** to save any files that may be replaced as part of the installation, or **No** if you do not want to make a backup.

If you select **Yes** the following dialog box allows you to specify the backup directory:

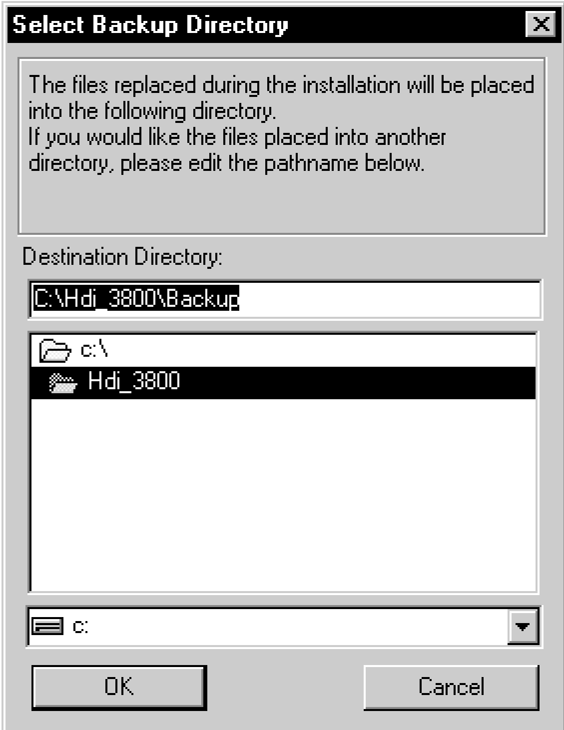


Figure 2.9 Select Backup Directory Dialog Box

Note: If there are no files to backup, a backup directory is not created even if the directory is specified.

- Enter the directory to be used and click **OK**.

The installer then copies the HDI files to the specified directory:

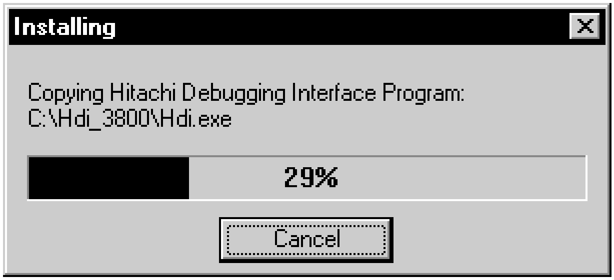


Figure 2.10 HDI Installing (1)

When first disk (#1/3) installation is completed, the following message is displayed. Then, insert installation disk #2/3 and press the **OK** button:

Figure 2.11 Insert New Disk Dialog Box (1)

The installer then copies the HDI file to the specified directory.

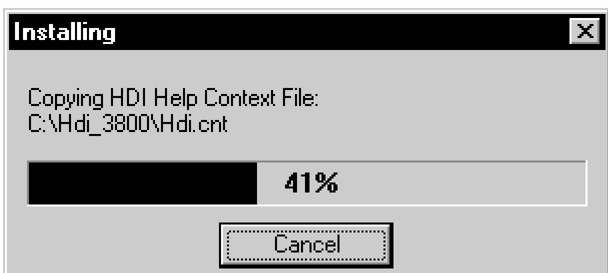


Figure 2.12 HDI Installing (2)

When second disk (#2/3) installation is completed, the following message is displayed. Then, insert installation disk #3/3 and press the **OK** button:

Figure 2.13   Insert New Disk Dialog Box (2)

Select the host interface type to be used in the following dialog box.

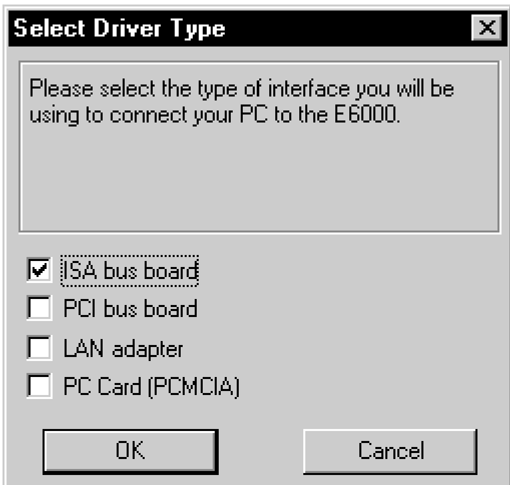


Figure 2.14   Select Driver Type Dialog Box

Finally this dialog box allows you to specify the program group for the HDI icon start menu:

Figure 2.15 Window for Specifying Program Groups of Icons

- Select an existing group or enter the name of a new group, and click **OK** to proceed.

2.4.1 Installation Details

The installer creates the following icons in the start menu you specified, by default HDI:

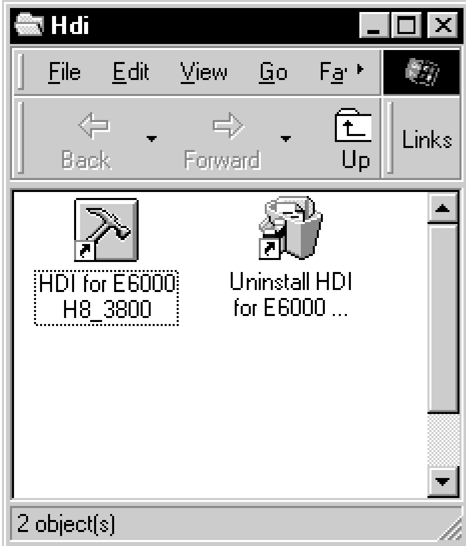


Figure 2.16   HDI Program Group

These icons have the following functions:

HDI for E6000 H8_3800 is the HDI program.

Uninstall HDI for E6000 H8_3800 will remove HDI, and its associated files, if you need to uninstall it.

2.5 Checking the System

The next step is to run the HDI software to check that the E6000 emulator is working correctly.

- Switch on the E6000 emulator and check that the red LED is illuminated.
- Select **HDI for E6000 H8_3800** from the start menu.

RENESAS

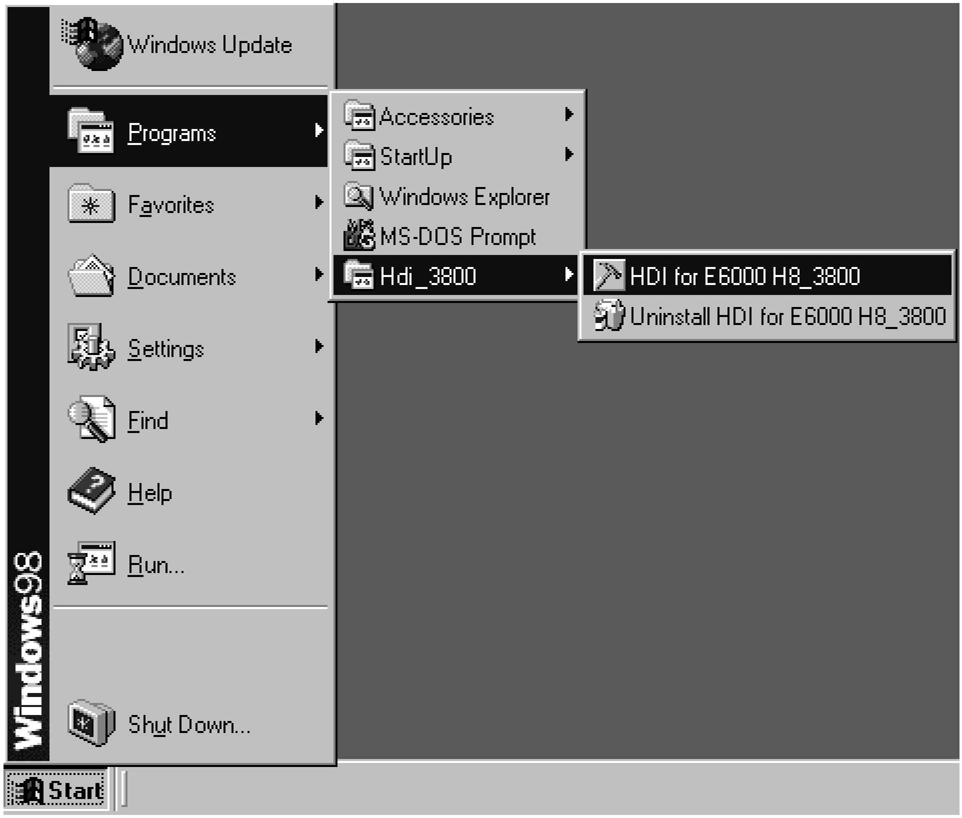


Figure 2.17 Start Menu

When the HDI window is displayed, the following messages are shown in the status bar under the window.

Figure 2.18 Status Bar during HDI Start-Up

Finally the status bar will display **Link up** to indicate that everything is set up correctly, and the HDI screen will be displayed as shown below.

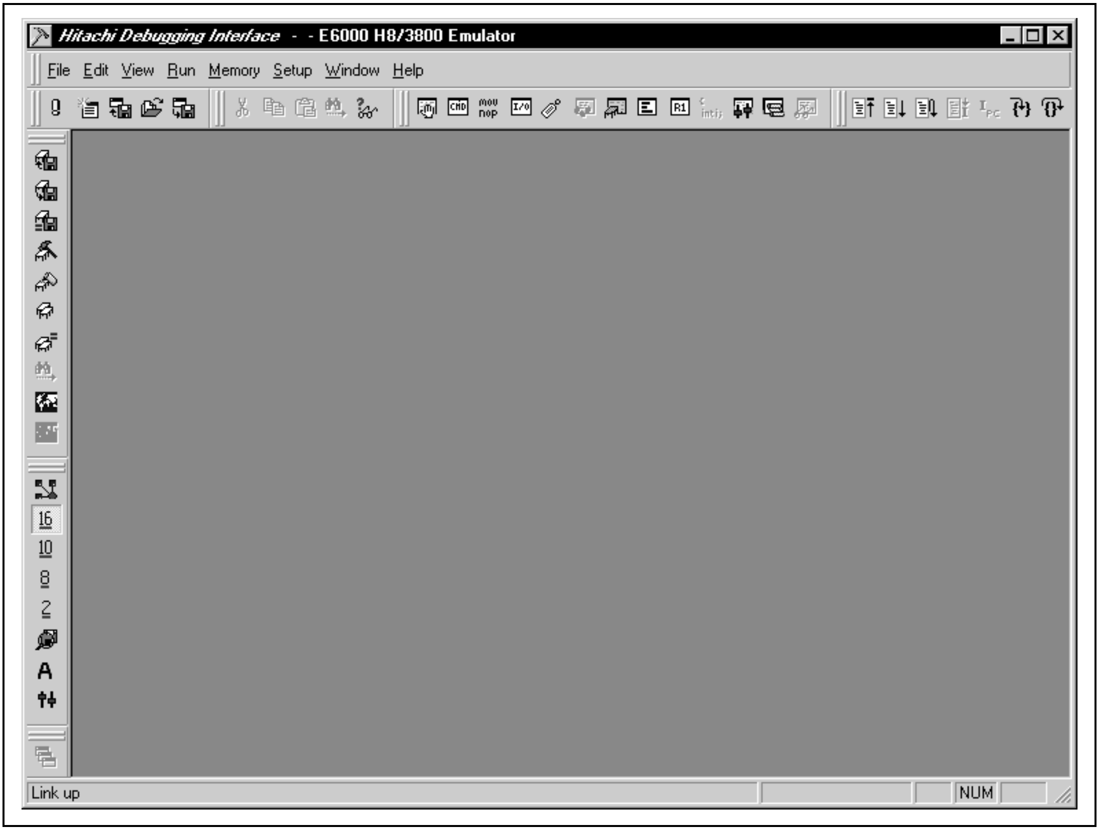


Figure 2.19 HDI Start-Up Screen

2.6 What Next?

The E6000 emulator is now correctly set up and ready for use. We recommend you work through section 4, Tutorial, to familiarize yourself with the key features of the E6000 emulator, and to learn how to use the E6000 emulator to develop and debug programs for MCU.

2.7 Uninstalling the HDI Software

This section describes how to uninstall the HDI software on Windows® 98, for example.

- Select **Uninstall HDI for E6000 H8/3800** from the **Start** menu.

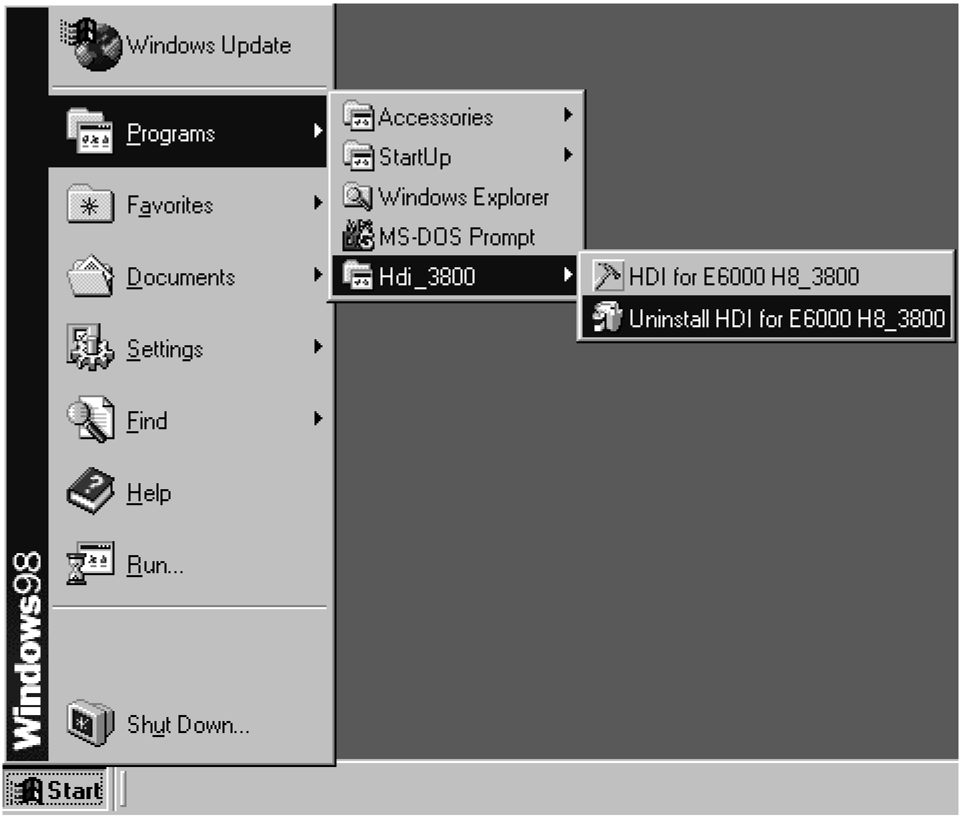


Figure 2.20 Start Menu (Uninstaller)

The uninstaller is initiated and the following dialog box will be displayed.

Figure 2.21   Select Uninstall Method Dialog Box

- To automatically uninstall the HDI, select the **Automatic** option button and click **Next**.
- To select the files to delete, select **Custom** option button and click **Next.**
- To cancel uninstallation, click **Cancel**.

When backup files were made at installation, the dialog box to confirm whether to roll back the backup files will be displayed.

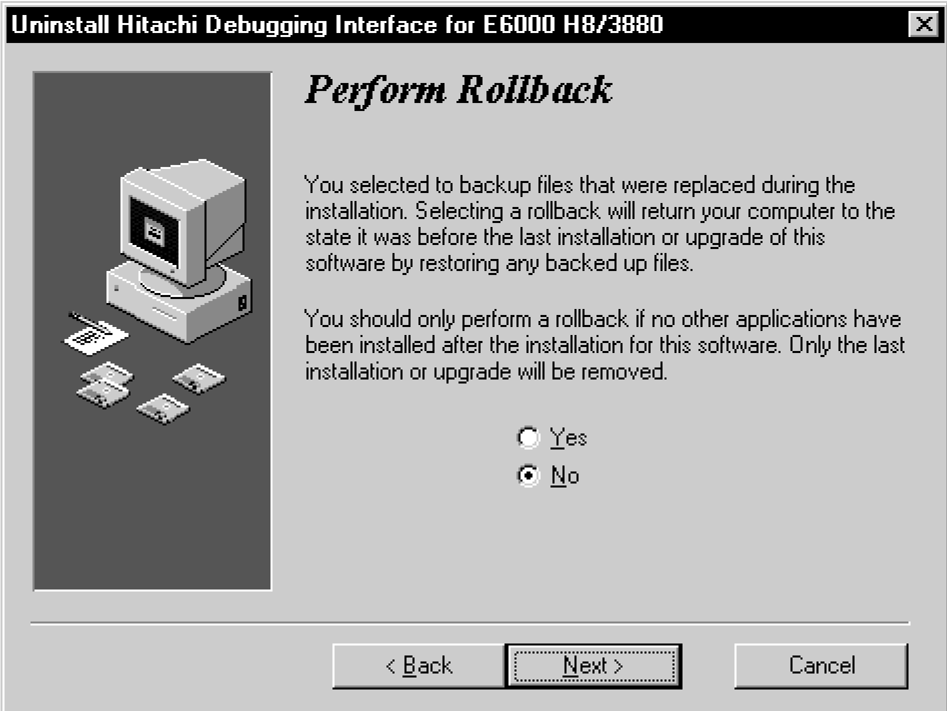


Figure 2.22   Perform Rollback Dialog Box

- To perform rollback, select the **Yes** option button and click **Next**.
- To not perform rollback, select the **No** option button and click **Next**.
- To cancel uninstallation, click **Cancel**.
- To go back to the **Select Uninstall Method** dialog box, click **Back**.

Notes: 1. By performing rollback, the backup files are restored.

2. If no backup files have been made or if no backup files are found, the **Perform Rollback** dialog box will not be displayed.

- The dialog box to confirm whether to start uninstallation will be displayed.

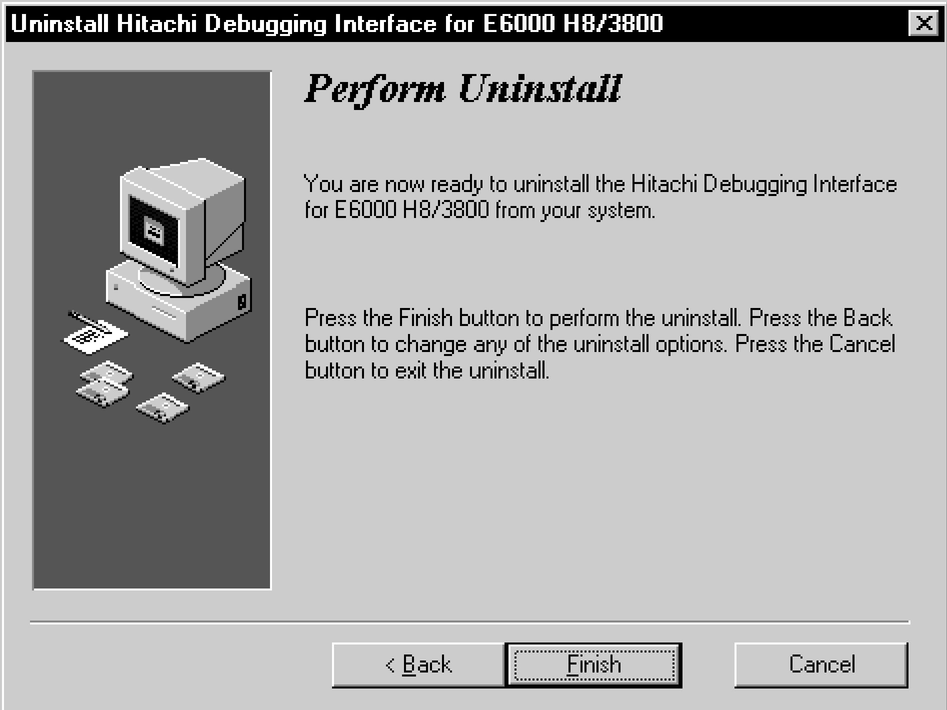


Figure 2.23 Perform Uninstall Dialog Box

- To start uninstallation, click **Finish**.
- To cancel uninstallation, click **Cancel**.
- To go back to the **Select Uninstall Method** dialog box, click **Back**.

When uninstallation is successfully completed, the directories and files created by the installer are deleted.

Note: 1. Any subdirectory or file that you have created in the HDI directory will not be deleted by the uninstaller.

2. When rollback was not performed, the backup directory and files will not be deleted.

2.8 Troubleshooting

2.8.1 Faulty Connection

If the PC interface board cannot detect the E6000 emulator, the following message box appears during initialization.

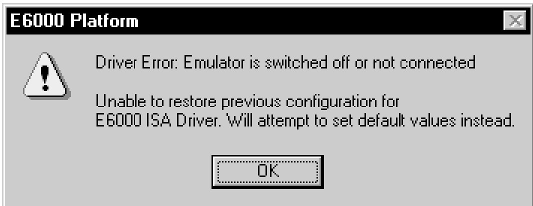


Figure 2.24 Faulty Connection Message (1)

The most likely reasons for this are:

- AC power supply adapter not connected to the E6000 emulator, or the emulator not switched on. Check the power LED on the E6000 emulator.
- The interface cable is not correctly connected between the PC interface board and the E6000 emulator.

2.8.2 Communication Problems

The following message indicates that the HDI was not able to set up the E6000 emulator correctly:

Figure 2.25 Faulty Connection Message (2)

This indicates;

- Incorrect area of memory reserved in the **CONFIG.SYS** file or interface switch incorrectly set on the rear panel of the PC interface board.
- Selected area of memory is in use by another application.

Check the setting according to section 2.2, Installing the PC Interface Board, and section 2.3, Setting Up the PC Interface Board on Windows NT® 4.0.

Section 3 Hardware

This section explains how to connect the E6000 emulator to user system.

3.1 Connecting to the User System

To connect the E6000 emulator to a user system proceed as follows:

- Connect the user system interface cable head to the user system.
- Plug the cable body into the E6000 emulator.
- Plug the cable body into the cable head.

For details of these steps, refer to the User System Interface Cable User's Manual.

Figure 3.1 gives details of the connectors provided on the E6000 emulator.

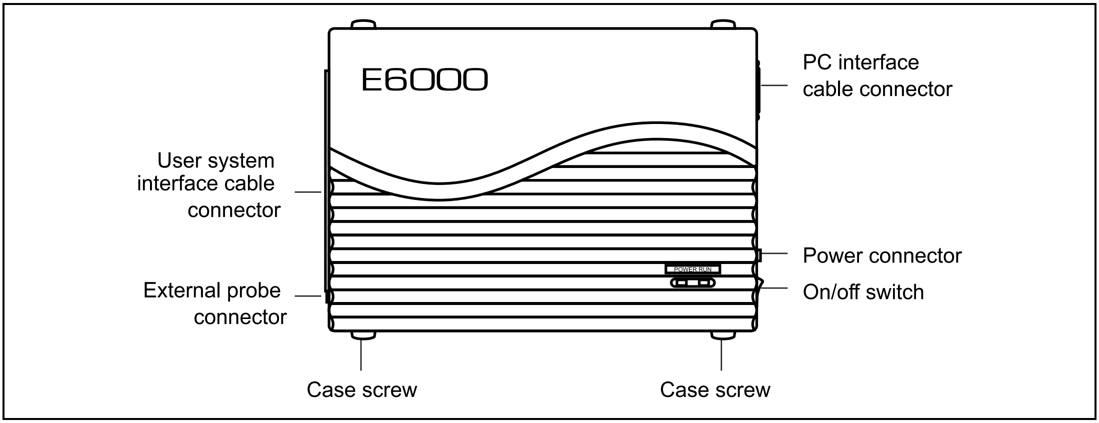


Figure 3.1 E6000 Emulator Connectors

3.1.1 Connecting Example of the User System Interface Cable Head to the User System

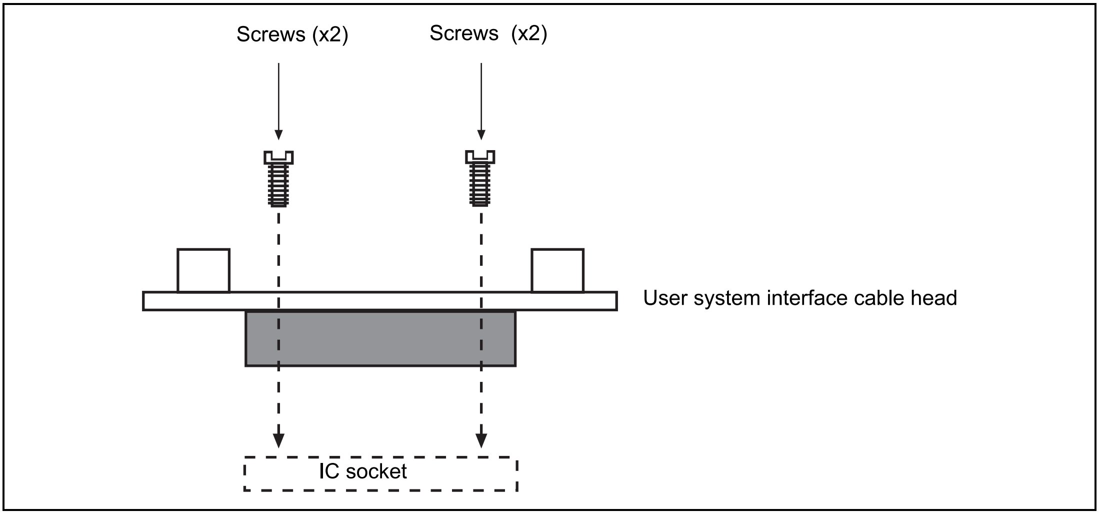


Figure 3.2 Connecting User System Interface Cable Head to User System

- Ensure that all power is off to the E6000 emulator, user system, and associated equipment.
- Insert the user system interface cable head into the socket on the user system.

Note: Depending upon the QFP package it may be possible to orientate this cable head in any position on the socket, so care should be taken to correctly identify pin 1 on the E6000 emulator part and socket when installing.

- Screw the user system interface cable head to the socket with the screws provided together with the user system interface cable. Alternately tighten the screws little by little in the sequence shown in figure 3.3.

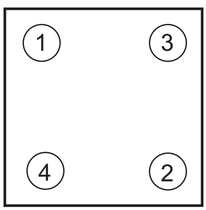


Figure 3.3 Sequence of Screw Tightening

Note: Be careful not to over-tighten the screws as this may result in contact failure on the user system hardware or damage the cable head. If the ‘solder lugs’ are provided on the QFP socket, use them to provide extra strength to the E6000 emulator and user system connection.

3.1.2 Plugging the User System Interface Cable Body into the E6000 Emulator

Plug the user system interface cable body into the E6000 emulator, taking care to insert it straight, and push it firmly into place.

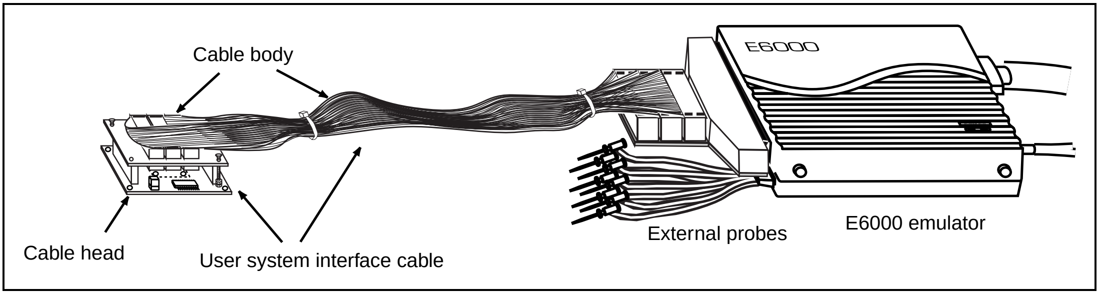


Figure 3.4 Plugging User System Interface Cable Body to E6000 Emulator

3.1.3 Plugging the User System Interface Cable Body into the User System Interface Cable Head

Plug the user system interface cable body into the user system interface cable head on the user system hardware.

3.2 Power Supply

3.2.1 AC Power-Supply Adapter

The AC adapter supplied with the E6000 emulator must be used at all times.

3.2.2 Polarity

Figure 3.5 shows the polarity of the power-supply plug.

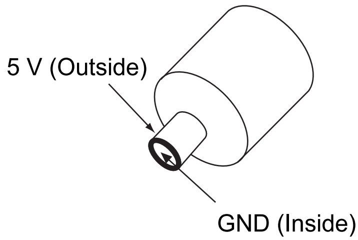


Figure 3.5 Polarity of Power Supply Plug

3.2.3 Power Supply Monitor Circuit

The E6000 emulator incorporates a user-system power supply monitor circuit which only lights the red LED when a voltage higher than 4.75 V is supplied. If this LED does not light, you should check the E6000 emulator voltage level. An input voltage less than 4.75 V could indicate that sufficient power is not supplied to the E6000 emulator.

Note: Use the provided AC power-supply adapter for the E6000 emulator.

3.3 Hardware Interface

All user system interface signals of the E6000 emulator are directly connected to the evaluation chip on the E6000 emulator with no buffering.

3.3.1 Signal Protection

All user system interface signals are protected from over- or under-voltage by use of diode arrays except for the AVcc and analog port signals.

Pull-up resistors are connected to the port signals except for the analog port signals.

The E6000 monitors the signals at the head of the user system interface cable to detect whether the user system hardware is connected.

3.3.2 User System Interface Circuits

The circuits that interface the evaluation chip on the E6000 emulator to the user system include pull-up resistors that cause signal delays. Note that when an input pin is in the high-impedance state, the pull-up resistor forces the pin to be at a high level. Adjust the user system hardware to compensate for these effects. The delay caused by the user system interface cable is about 3 ns.

The following diagrams show the user system interface signal circuits.

Signals Other than below:

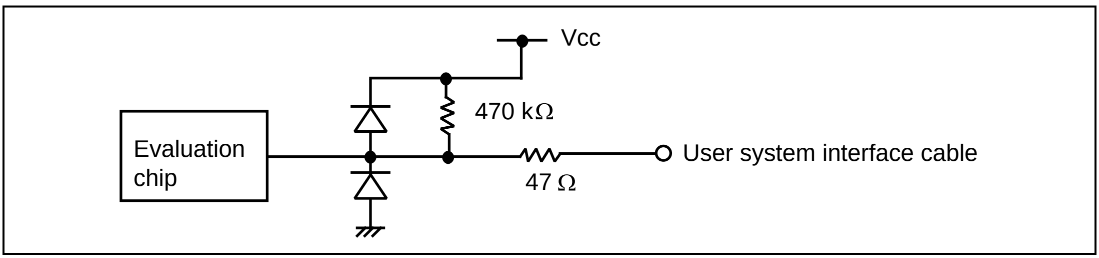


Figure 3.6 User System Interface Circuit for Other Signals

OSC1 and X1:

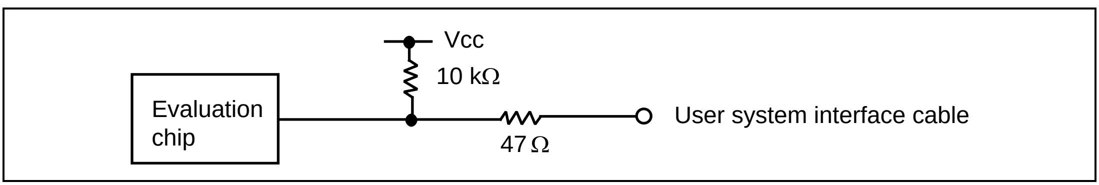


Figure 3.7 User System Interface Circuit for OSC1 and X1

P50/$\overline{\text{WKP0}}$/SEG1 to P57/$\overline{\text{WKP7}}$/SEG8, P60/SEG9 to P67/SEG16, P70/SEG17 to P77/SEG24, P80/SEG25 to P87/SEG32/CL1, PC0/COMP0 to PC3/COMP3, PB0/AN0 to PB7/AN7:

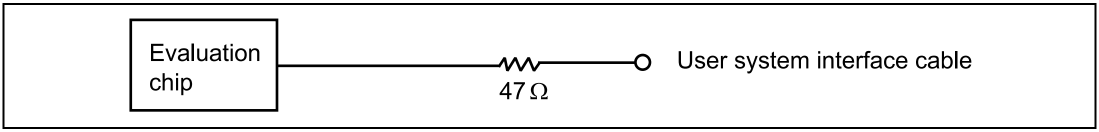


Figure 3.8 User System Interface Circuit for P50/$\overline{\text{WKP0}}$/SEG1 to P57/$\overline{\text{WKP7}}$/SEG8, P60/SEG9 to P67/SEG16, P70/SEG17 to P77/SEG24, P80/SEG25 to P87/SEG32/CL1, PC0/COMP0 to PC3/COMP3, PB0/AN0 to PB7/AN7

AVcc and AVss:

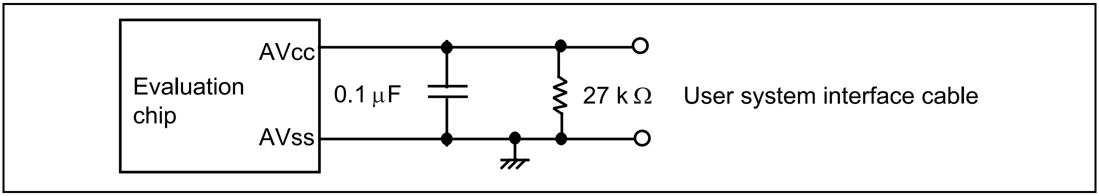


Figure 3.9 User System Interface Circuit for AVcc and AVss

CVcc and TEST:

When CVcc is connected to GND, or TEST is connected to Vcc level, a warning message is displayed at HDI initiation. Check the CVcc and TEST pins on the user system.

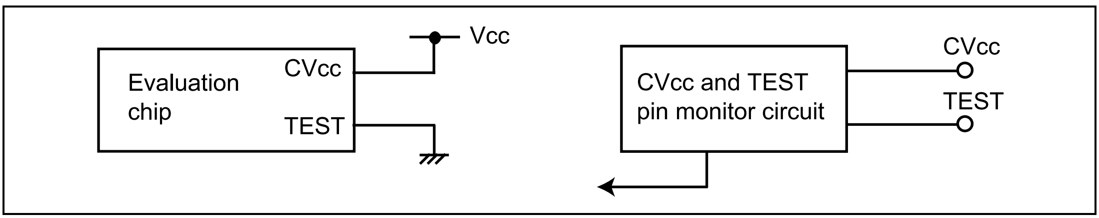


Figure 3.10 User System Interface Circuit for CVcc and TEST

V0, V1, V2, and V3:

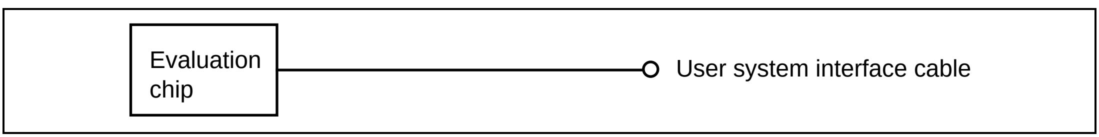


Figure 3.11 User System Interface Circuit for V0, V1, V2, and V3

3.3.3 Clock Oscillator

Figure 3.12 shows the system clock oscillator on the user system interface cable. This oscillator is designed to oscillate in the range of 1 MHz to 16 MHz. For details, refer to the user system interface cable manuals.

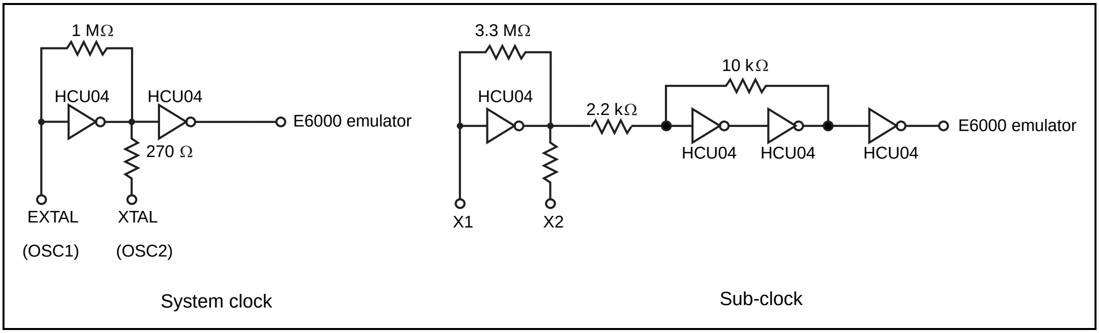


Figure 3.12 Oscillator Circuit

3.3.4 External Probes/Trigger Output

An 8-pin connector, marked EXT (next to the user interface connector), on the E6000 emulator case accommodates four external probe inputs and two trigger outputs. The pinout of this connector is shown in figure 3.13.

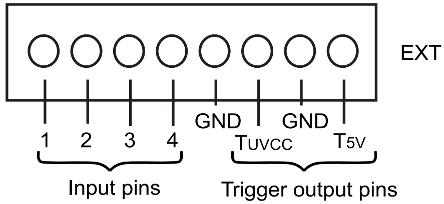


Figure 3.13 External Probe Connector

The external probe interface circuit is shown in figure 3.14.

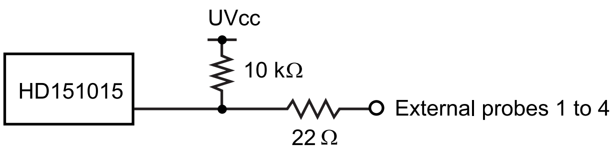


Figure 3.14 External Probe Interface Circuit

The trigger output is controlled by event channel 8 and is low active. The trigger output is available as either T5 V (probe color: white; within the range from 2.5 V to 5 V and does not depend on the user V_{CC} level) or TU_{VCC} (probe color: yellow; the user V_{CC} level). When the TU_{VCC} is used, user system cannot be evaluated at the power voltage of 1.8 V. (The voltage must be within the range 2.0 V to 5.0 V.)

3.3.5 Voltage Follower Circuit

A voltage follower circuit is implemented on the E6000 emulator which allows the user system voltage level from the user system to be monitored. This monitored voltage level is automatically supplied to the logic on the E6000 emulator and is derived from the E6000 emulator power supply unit. This means that no power is taken from the user system board.

If no user system interface cable is connected to the E6000 emulator, the E6000 emulator will operate at 5 V and all clock frequencies will be available to the user. If the user system interface cable is attached, the E6000 emulator will match the voltage supplied to the user system in all cases; i.e. even when the user V_{CC} is below the operating voltage for the MCU. You must be careful not to select an invalid clock frequency if operating at less than 5 V.

You can set a user V_{CC} threshold in the range 5 V to 0 V by using the E6000 emulator configuration dialog box. If the user V_{CC} drops below this threshold, the **User System Status** in the system status window will display **Down**; otherwise **OK** is printed. When the user system interface cable is disconnected, the E6000 emulator Vcc level is 5 V.

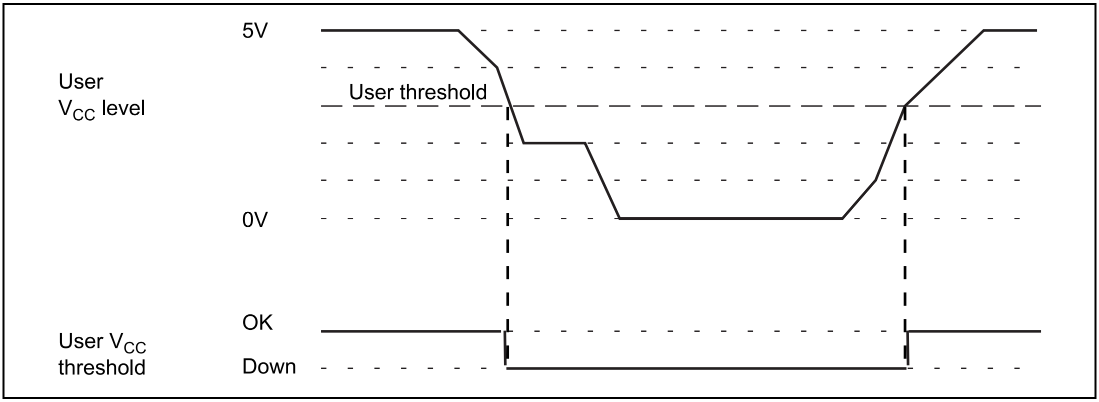


Figure 3.15 Voltage Level Monitoring

RENESAS

3.4 Differences between MCU and E6000 Emulator

When the E6000 emulator is initialized or the system is reset there are some differences in the initial values in some of the general registers as shown in table 3.1.

Table 3.1 Initial Value Differences between MCU and E6000 Emulator

Status	Register	E6000 Emulator	MCU
Power-on	PC	Undefined	Reset vector value
	R0 to R6	0000	Undefined
	R7 (SP)	0010	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined	The I mask is set to 1 and the other bits are undefined
Reset command	PC	Reset vector value	Reset vector value
	R0 to R6	Undefined	Undefined
	R7 (SP)	0010	Undefined
	CCR	The I mask is set to 1 and the other bits are undefined	The I mask is set to 1 and the other bits are undefined

Please refer to section 3.3, Hardware Interface, for details of the protection circuitry used on the I/O ports of the E6000 emulator.

3.4.1 A/D Converter

Due to the use of a user system interface cable there is a slight degradation in the A/D resolution and above that quoted in the Hardware Manual for the MCU being emulated.

3.4.2 Access to Unused Area

The unused area from H'FF80 to H'FF8F is used by the emulator system. Therefore, if this area is allocated to the emulator by the MAP setting, the operation is not guaranteed. Do not use this area.

3.4.3 Program Execution by the Go Reset Command

When the program is executed using the Go Reset command, the E6000 emulator inputs a 500-μs reset signal to the evaluation chip. This reset signal input time is added when the execution time measurement result is displayed.

Section 4 Tutorial

The following describes a sample debugging session, designed to introduce the main features of the E6000 emulator used in conjunction with the Hitachi debugging interface (HDI) software.

The tutorial is designed to run in the E6000 emulator's resident memory so that it can be used without connecting the E6000 emulator to an external user system.

4.1 Introduction

The tutorial is based on a simple C program.

Before reading this chapter:

- Set up the E6000 emulator and verify that it is working in conjunction with the HDI software by referring to section 2, Setting Up. You do not need to connect the E6000 emulator to a user system to use this tutorial.
- Make sure you are familiar with the architecture and instruction set of the MCU. For more information refer to the H8/3802 series Hardware Manual.

4.1.1 Overview

This program is an infinite loop that sorts elements based on NAME in the alphabetical order, and AGE and ID in the ascending order.

The tutorial is provided on the installation disk as the file **tutorial.c**. A compiled version of the tutorial is provided in Sysrof format in the file **tutorial.abs** on the installation disk.

4.2 How the Tutorial Program Works

The first part of the program includes a series of header files:

```
#include <machine.h>
#include "string.h"
```

The program then gives prototypes for the constants, structures, and function initial values:

```
#define NAME    (short)0
#define AGE     (short)1
#define ID      (short)2
#define LENGTH  8

struct namelist {
   char    name[LENGTH];
   short   age;
   long    idcode;
};
struct namelist section1[] = {
   "Naoko",  17, 1234,
   "Midori", 22, 8888,
   "Rie",    19, 7777,
   "Eri",    20, 9999,
   "Kyoko",  26, 3333,
   "",        0,    0
};

int count;

void sort();
```

Now the **main** function.

```
main( )
{
   count = 0;
   for ( ; ; ){
      sort(section1, NAME);
```

```
            count++;
            sort(section1, AGE);
            count++;
            sort(section1, ID);
            count++;
        }
    }
```

The remainder of the program defines the functions called from **main**:

```
    void sort(list, key)
    struct namelist list[];
    short key;
    {
        short i,j,k;
        long min;
        char *name;
        struct namelist worklist;

        switch(key){
            case NAME :
                for (i = 0 ; *list[i].name != 0 ; i++){
                    name = list[i].name;
                    k = i;
                    for (j = i+1 ; *list[j].name != 0 ; j++){
                        if (strcmp(list[j].name , name) < 0){
                            name = list[j].name;
                            k = j;
                        }
                    }
                    worklist = list[i];
                    list[i] = list[k];
                    list[k] = worklist;
                }
                break;

            case AGE  :
                for (i = 0 ; list[i].age != 0 ; i++){
                    min = list[i].age;
```

```
                k = i;
                for (j = i+1 ; list[j].age != 0 ; j++){
                    if (list[j].age < min){
                        min = list[j].age;
                        k = j;
                    }
                }
                worklist = list[i];
                list[i] = list[k];
                list[k] = worklist;
            }
            break;

        case ID   :
            for (i = 0 ; list[i].idcode != 0 ; i++){
                min = list[i].idcode;
                k = i;
                for (j = i+1 ; list[j].idcode != 0 ; j++){
                    if (list[j].idcode < min){
                        min = list[j].idcode;
                        k = j;
                    }
                }
                worklist = list[i];
                list[i] = list[k];
                list[k] = worklist;
            }
            break;
    }

}
```

4.3 Running HDI

To run the HDI:

- Select **HDI for E6000 H8_3800** from the start menu:

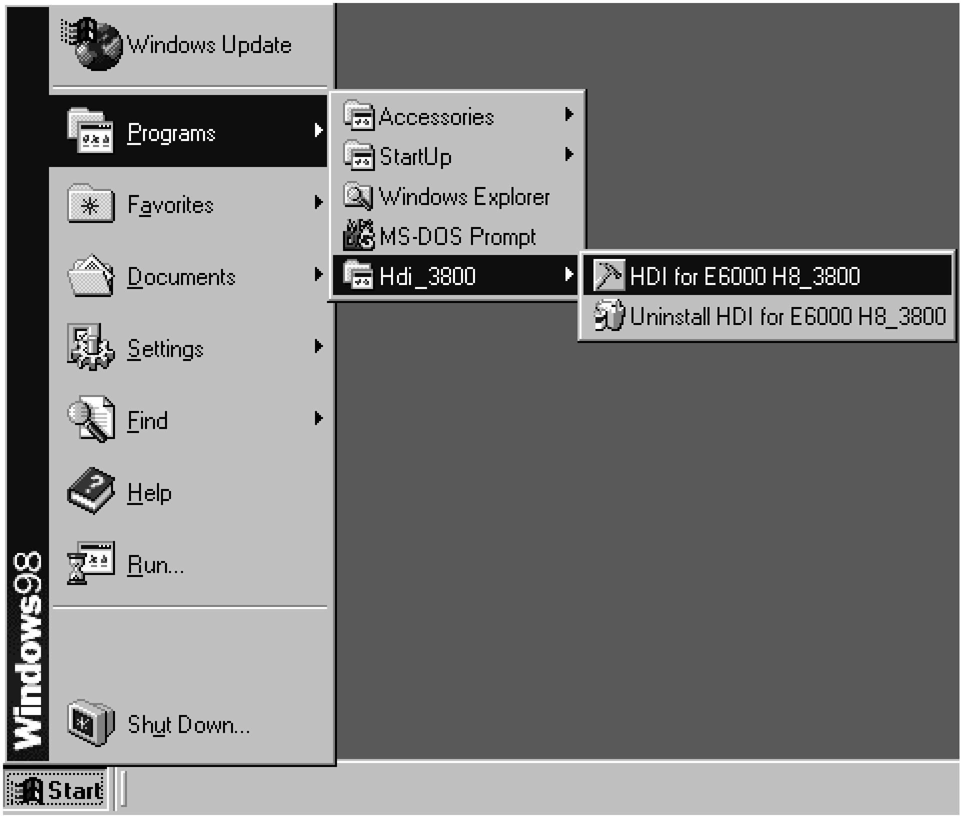


Figure 4.1 Start Menu

4.3.1 Selecting the Target Platform

The HDI has extended functions for supporting multiple target platforms, and if your system is set up for more than one platform you will first be prompted to choose a platform to be used.

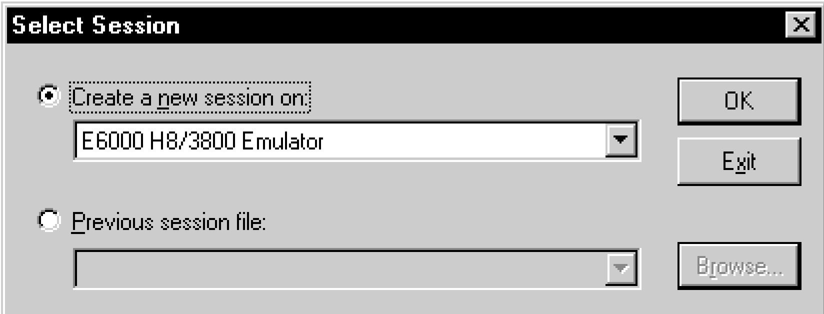


Figure 4.2 Select Platform Dialog Box

- For this tutorial select **E6000 H8/3800 Emulator** and click **OK** to continue.

Note that you can change the target platform at any time by choosing **New Session…** from the **File** menu.

When the emulator has been successfully set up the **Hitachi Debugging Interface** window will be displayed, with the message **Link up** in the status bar. Figure 4.3 shows the key features of the window.

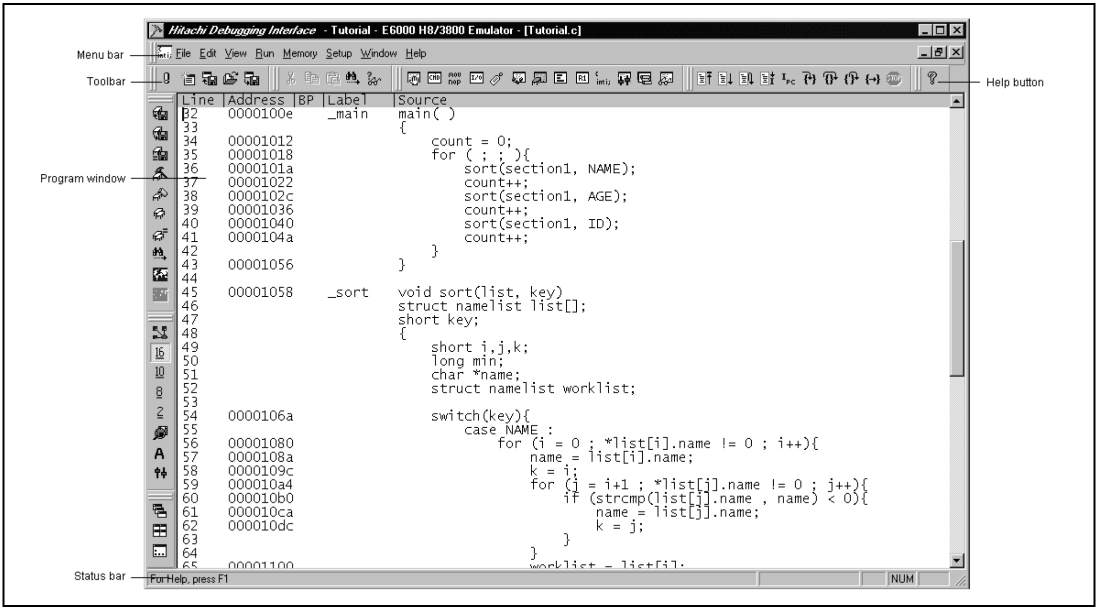


Figure 4.3 Hitachi Debugging Interface Window

For details on the HDI key functions, refer to the Hitachi Debugging Interface User's Manual.

4.3.2 Menu

The menu bar gives you access to the HDI commands for setting up the E6000 emulator and using the HDI debugging functions.

Toolbar: Provides convenient buttons as shortcuts for the most frequently used menu commands.

Program Window: Displays the source of the program being debugged.

Status Bar: Displays the status of the E6000 emulator. For example, progress information about downloads, snapshots of address bus in run mode.

Help Button: Activates context sensitive help about any feature of the HDI user interface.

4.4 Setting up the E6000 Emulator

Before downloading a program to the E6000 emulator you first need to set up the target MCU conditions. The following items need to be configured:

- The device type
- The operating mode
- The clock source
- The user signals
- The memory map

The following sections describe how to set up the E6000 emulator as appropriate for the tutorial program.

4.4.1 Configuring the Platform

To set up the target configuration.

- Choose **Configure Platform…** from the **Setup** menu.
- The following dialog box will be displayed:

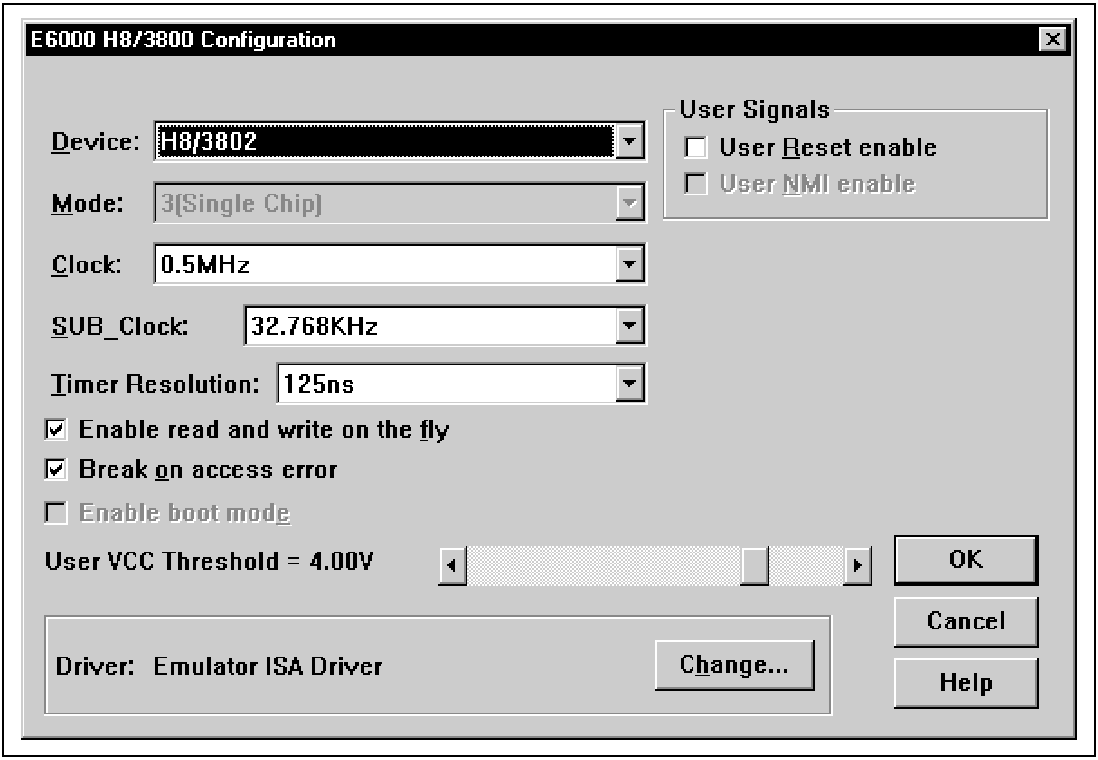


Figure 4.4 Target Configuration Dialog Box

- Set up the options as shown in table 4.1

Table 4.1 Target Configuration Options

Option	Value
Device	H8/3802
Mode	3 (single chip) cannot be changed
Clock	0.5 MHz
Timer resolution	125 ns
User VCC level (threshold)	4.00 V
All other options	Enabled

- Click **OK** to change the target configuration.

4.4.2 Mapping the Memory

The HDI automatically maps the E6000 emulator memory according to the device and mode set in the **Configuration** dialog box.

- To display the current memory mapping, choose **Configure Map** from the **Memory** menu, or click the **Memory Map** button in the toolbar.

The **Memory Mapping** dialog box shown in figure 4.5 is displayed.

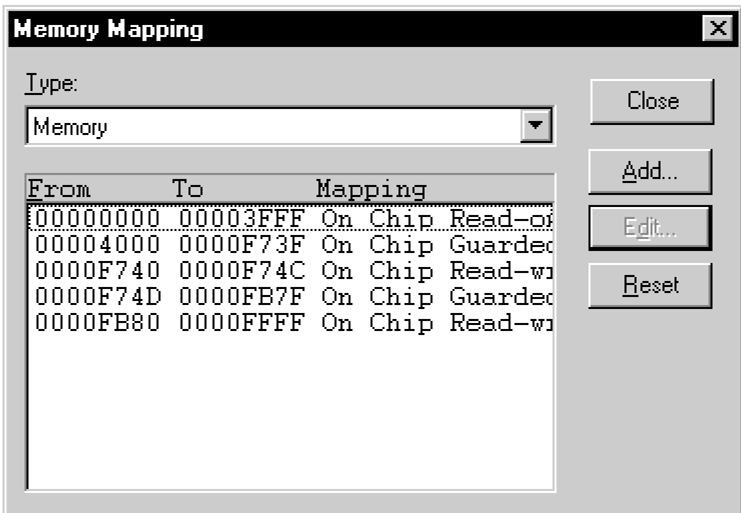


Figure 4.5 Memory Mapping Dialog Box

Table 4.2 lists the two memory types available in the E6000 emulator.

Table 4.2 Memory Type

Memory Type	Description
Internal	Accesses the MCU internal memory.
Emulator	Accesses the emulation memory.

Table 4.3 lists the three access types.

Table 4.3 Access Types

Access Type	Description
Read-write	RAM.
Read-only	ROM.
Guarded	No access allowed.

For this tutorial we can use the default mapping, but you can edit the mapping as follows:

- To edit the mapping, select the appropriate map setting value and click the **Edit** button, or double-click the line of the appropriate map setting. Double-click the **Internal Read-only** in the **Memory Mapping** dialog box.

The **Edit Memory Mapping** dialog box is displayed.

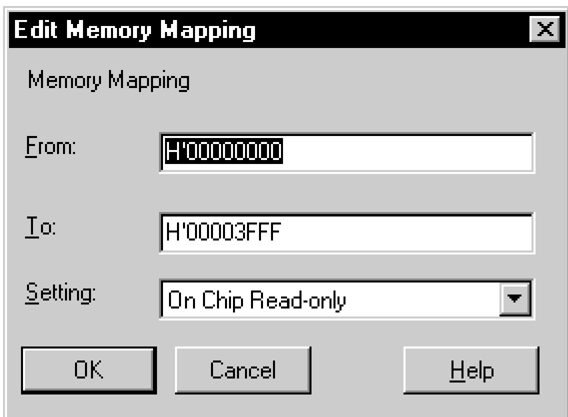


Figure 4.6 Edit Memory Mapping Dialog Box

- Click **OK** to close the dialog box.
- To display the device map information, select **Status** from the **View** menu or click the **Status** button in the toolbar to open the **System Status** window, and select the **Memory** panel. The device map information is then displayed as follows:

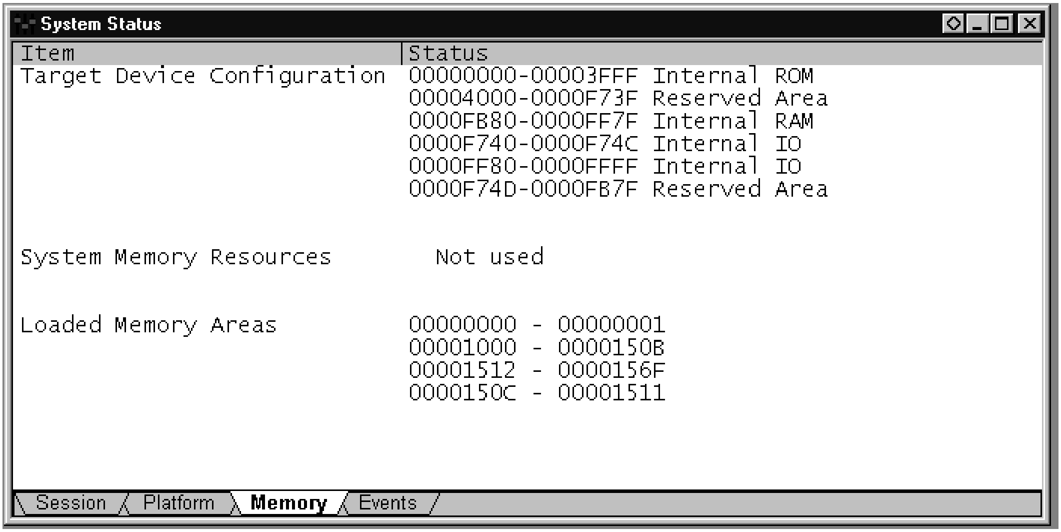


Figure 4.7 System Status Window (Memory Panel)

Note: Memory mapping differs according to the target MCU.

4.5 Downloading the Tutorial Program

Once the E6000 emulator is set up you can download the object program you want to debug.

4.5.1 Loading the Object File

First load the Sysrof-format object file, as follows:

- Choose **Load Program…** from the **File** menu, or click the **Load Program** button in the toolbar.

  The **Load Program** dialog box is opened.

- Click the **Browse...** button, select the **Tutorial.abs** file in the **Tutorial** directory from the **Open** dialog box, and click the **Open** button. The **Load Program** dialog box is displayed. Click the **Open** button to start to download the file.

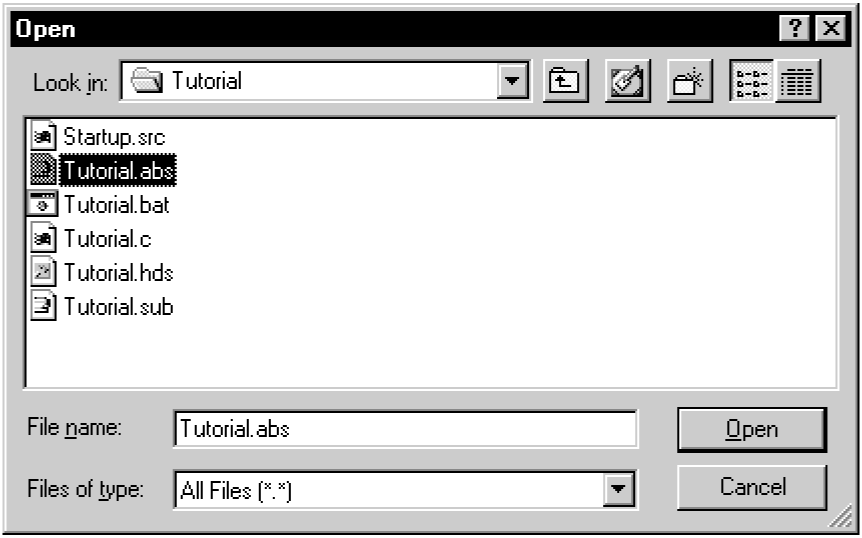


Figure 4.8 Open Dialog Box (Selecting the Object File)

When the file has been loaded the dialog box shown in figure 4.9 displays information about the memory areas that have been filled with the program code.

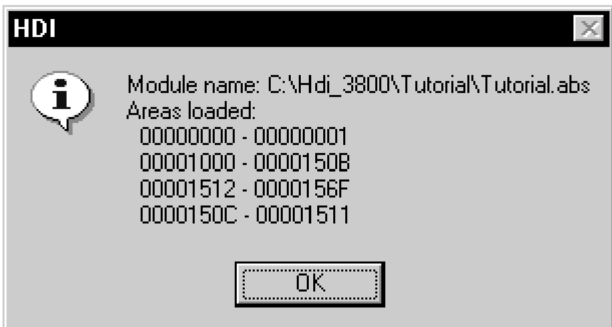


Figure 4.9 HDI Information Message Box

- Click **OK** to continue.

Note that all the code lies within the internal ROM.

4.5.2 Displaying the Program Listing

HDI allows you to view a program at source level and in assembly-language mnemonic.

- Choose **Source…** from the **View** menu, or click the **Program Source** button in the toolbar.

You will be prompted for the C source file corresponding to the object file you have loaded.

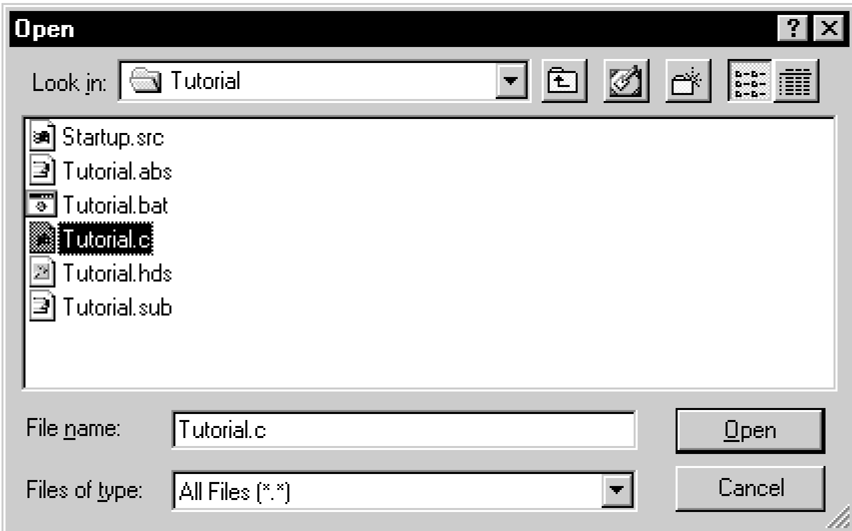


Figure 4.10 Open Dialog Box (Selecting a Source File)

- Select **Tutorial.c** and click **Open** to display the program window.

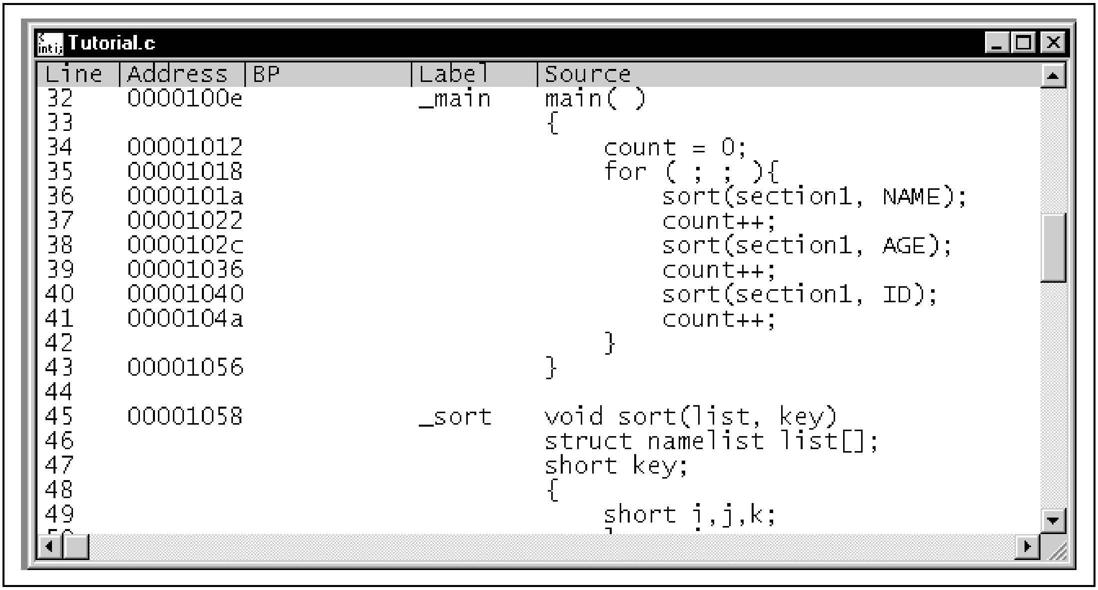


Figure 4.11   Tutorial Program Window

- If necessary choose **Font…** option from the **Customize** submenu on the **Setup** menu to choose a font and size suitable for your host computer.

Initially the program window opens showing the start of the main program, but you can scroll through the program with the scroll bars to see the definitions and include statements.

4.6 Using Breakpoints

The simplest debugging aid is the program breakpoint, which lets you halt execution when a particular point in the program is reached. You can then examine the state of the MCU and memory at that point in the program.

4.6.1 Setting a Program Breakpoint

The program window provides a very simple way of setting a program breakpoint. For example, set a breakpoint at address **H'1036** as follows:

- Double-click in the **BP** column on the line containing address **H'1036**.

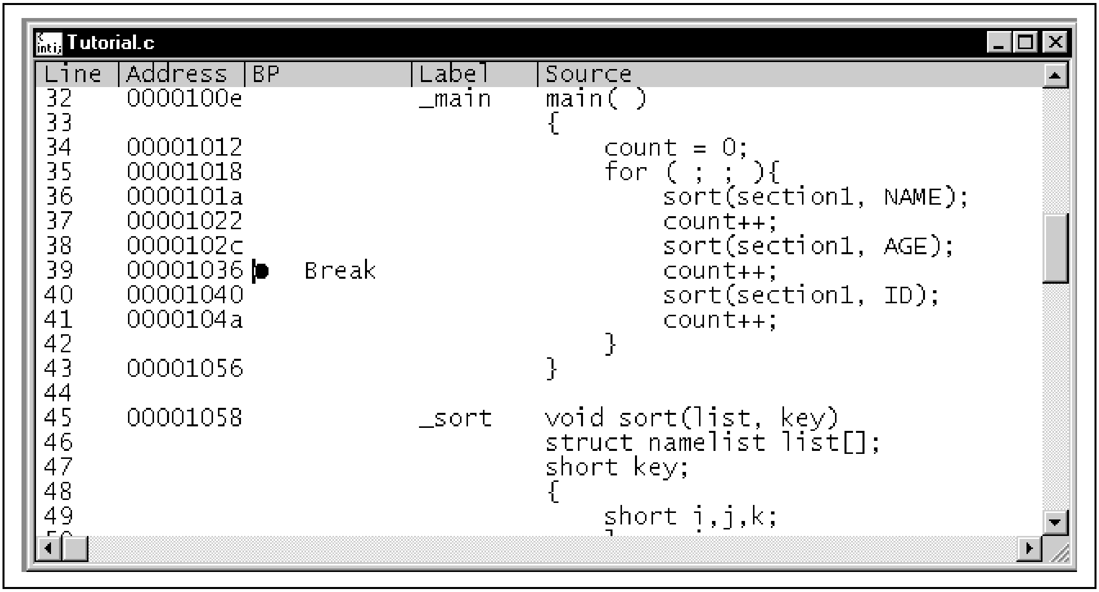


Figure 4.12 Setting a Breakpoint

The word • **Break** will be displayed there to show that a program breakpoint is set at that address. Although not performed in this tutorial, double-clicking repeatedly in the **Break** column can change the display in the cyclic order shown below to set the event for measuring the execution time between events (**+Timer**: start measurement; **-Timer**: stop measurement), point-to-point trace (**+Trace**: start trace; **-Trace**: stop trace), or trace stop (**TrStop**: trace stop).

(Blank) → **Break** → **+Timer** → **-Timer** → **+Trace** → **-Trace** → **TrStop** → (Blank) → ...

or

-Trace

4.6.2 Executing the Program

To run the program from the address pointed to by the reset vector:

- Choose **Reset Go** from the **Run** menu, or click the **Reset Go** button in the toolbar.

The program will be executed up to the breakpoint you inserted, and the statement will be highlighted in the program window to show that the program has halted.

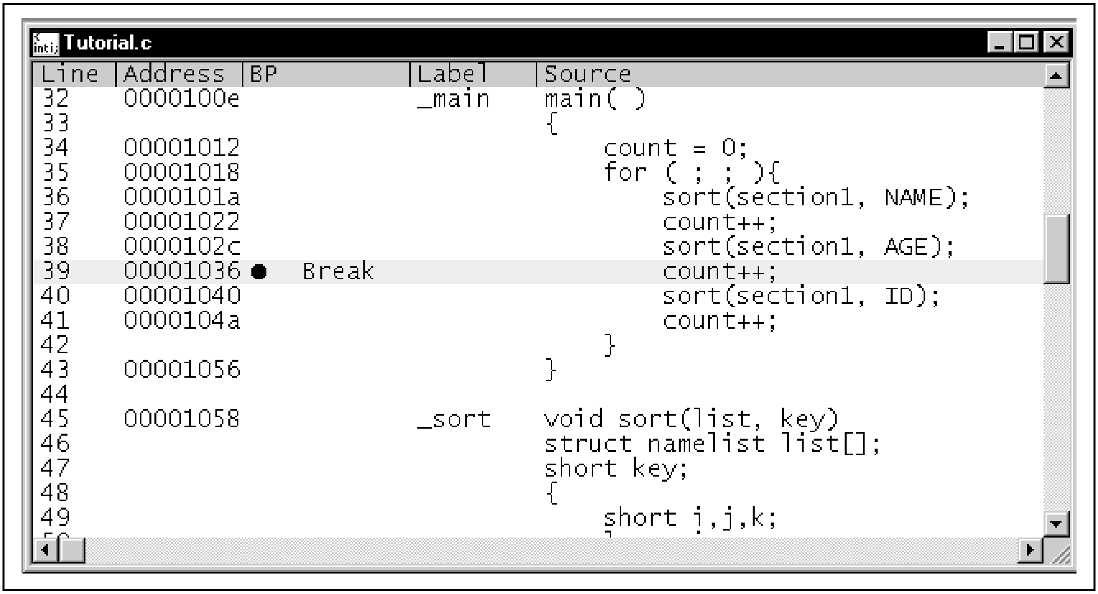


Figure 4.13 Program Break

The message **Break=Soft Ware Breakpoint** is displayed in the status bar to show the cause of the break.

You can also see the cause of the last break in the **System Status** window.

- Choose **Status** from the **View** menu, or click the **Status** button in the toolbar to open the **System Status** window and choose the **Platform** panel:

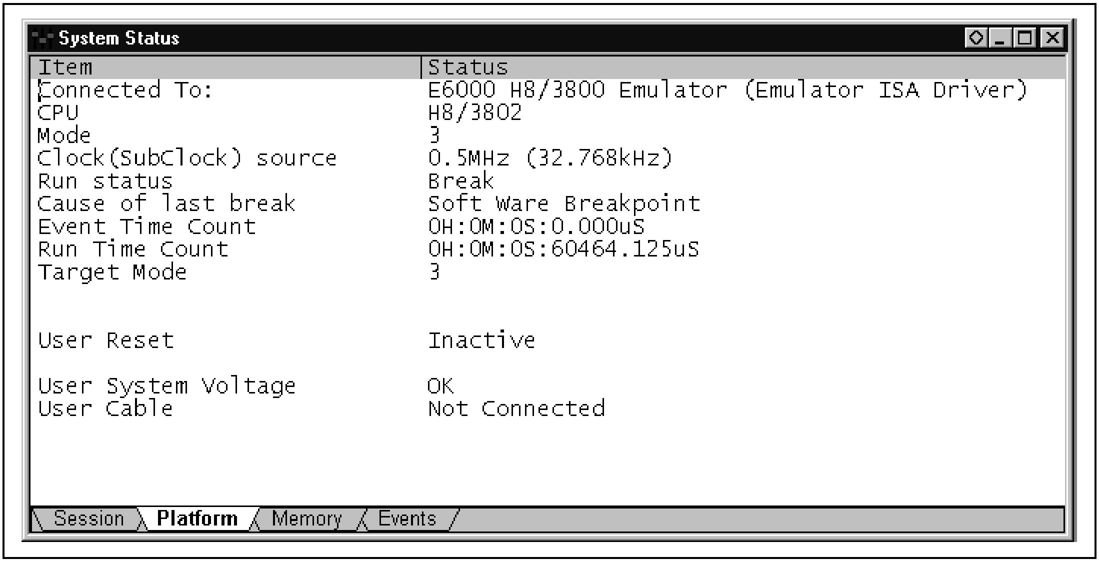


Figure 4.14 System Status Window (Platform panel)

The **Cause of last break** line shows that the break was a program break.

4.6.3 Examining Registers

While the program is halted you can examine the contents of the MCU registers. These are displayed in the **Registers** window.

- Choose **Registers** from the **View** menu, or click the **CPU Registers** button in the toolbar:

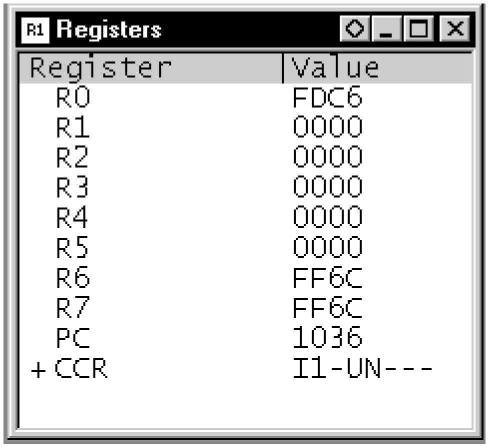


Figure 4.15 Registers Window

As expected, the value of the program counter, PC, is the same as the highlighted statement, **H'1036**. (Note that other register values may be different from ones shown in figure 4.14.)

You can also change the registers from the **Registers** window. For example, to change the value of the PC:

- Double-click the **Value** column corresponding to **PC** in the **Registers** window.

The **Register-PC** dialog box allows you to edit the value.

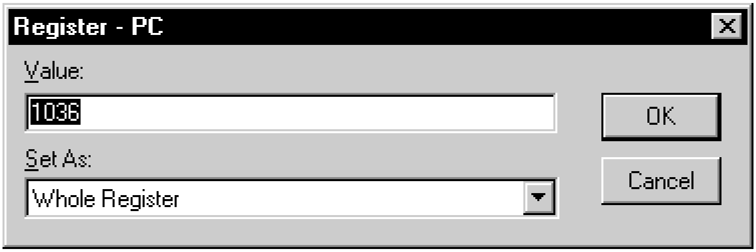


Figure 4.16   Changing Register Value

- Edit the value to **H'102C**, the address of the previous statement, and click **OK**.

The highlighted bar will move to the previous statement in the program window to show the new program counter value.

- Choose **Go** from the **Run** menu, or double-click the **Go** button in the toolbar, to execute up to the breakpoint again.

4.6.4 Reviewing the Breakpoints

You can see a list of all the breakpoints set in the program in the Breakpoints window.

- Choose **Breakpoints** from the **View** menu, or click the **Breakpoints** button in the toolbar:

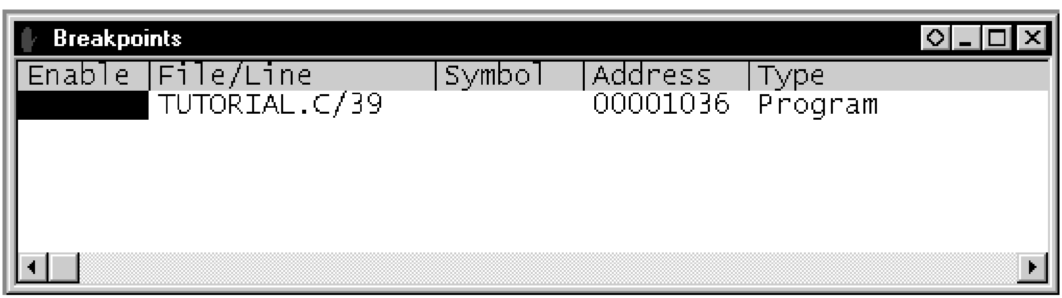


Figure 4.17   Breakpoints Window

The **Breakpoints** window also allows you to enable and disable breakpoints, define new breakpoints, and delete breakpoints.

RENESAS

Before proceeding remove the breakpoint as follows:

- Highlight the breakpoint in the **Breakpoints** window and click **Delete**.
- Close the **Breakpoints** window.

4.7 Examining Memory and Variables

You can monitor the behavior of a program by examining the contents of an area of memory, or by displaying the values of variables used in the program.

4.7.1 Viewing Memory

You can view the contents of a block of memory in the **Memory** window.

For example, to view the memory corresponding to the structure **section1** in Byte:

- Choose **Memory…** from the **View** menu, or click the **Memory** button in the toolbar.

- Enter **section1** in the **Address** field, and set **Format** to **Byte**.

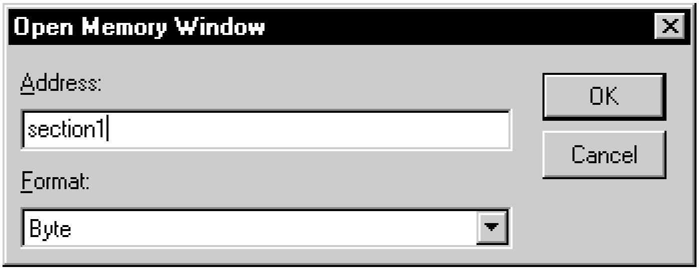


Figure 4.18 Open Memory Window

- Click **OK** to open the **Memory** window showing the specified area of memory.

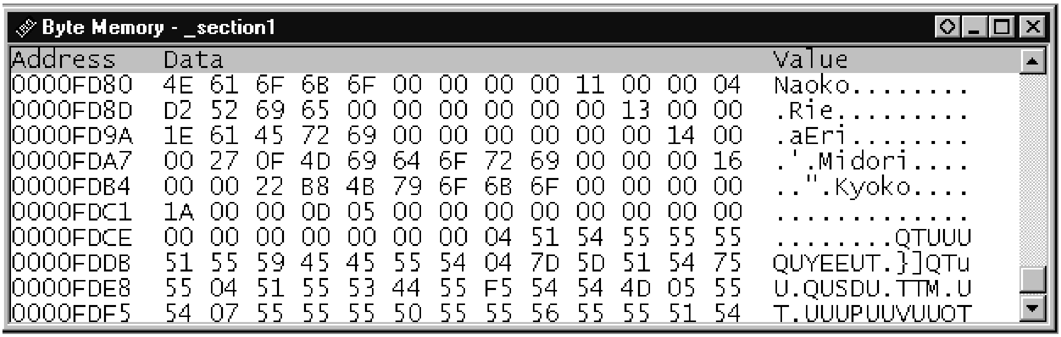


Figure 4.19 Memory Window (Byte)

- Leave the **Memory** window open so that you can monitor the contents of the structure **section1**.

4.7.2 Watching Variables

As you step through a program it is useful to be able to watch the values of variables used in your program, to verify that they change in the way that you expected.

For example, set a watch on the structure variable **section1**, declared at the beginning of the program, using the following procedure:

- Scroll up in the program window until you see the line:

  sort(section1, ID);

- Click to position the cursor to the left of **section1** in the program window.

- Click in the program window with the right button of the mouse to display a pop-up menu, and choose **Add Watch**.

A variable will be displayed in the **Watch** window.

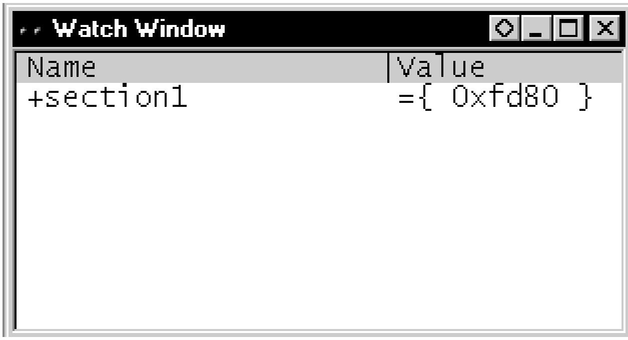


Figure 4.20 Watch Window

You can also add a watch to the **Watch** window by specifying its name. Use this method to add a watch on the variable **count** as follows:

You can double-click the **+** symbol to the left of symbol **section 1** in the **Watch** window to expand it and display the individual elements in the array.

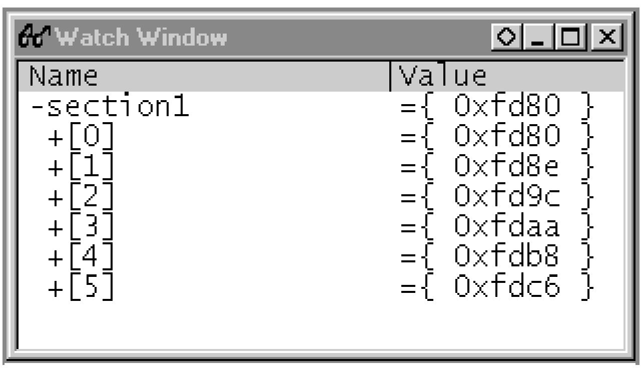


Figure 4.21 Watch Window (Symbol Extension)

A variable name can be specified to add a variable to the **Watch** window.

- Click in the **Watch** window with the right button of the mouse to display a pop-up menu, and choose **Add Watch...**.
- Enter variable name **count** and click the **OK** button.

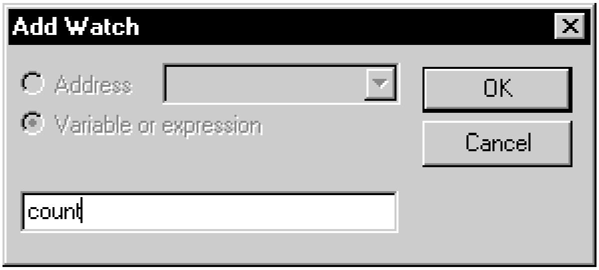


Figure 4.22 Add Watch Dialog Box

The int-type variable **count** is added to the **Watch** window.

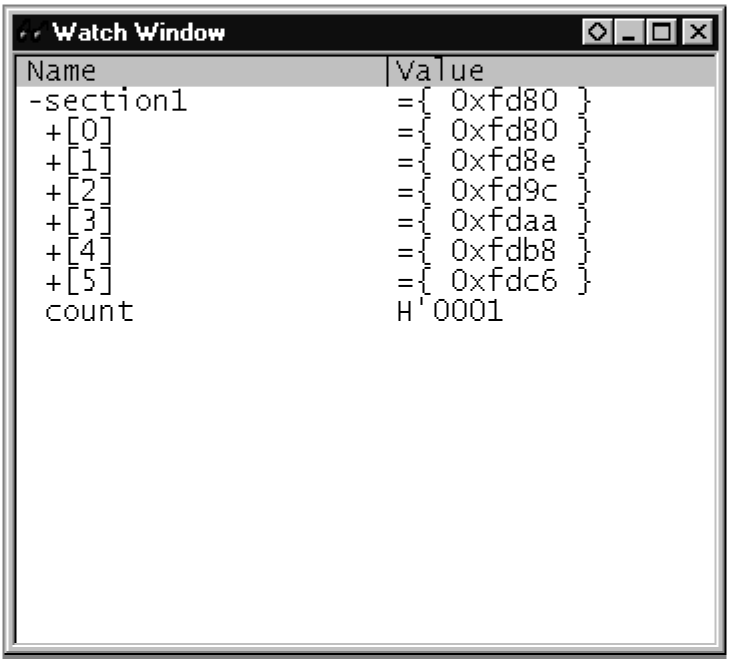


Figure 4.23 Watch Window (Adding Variables)

4.8 Stepping Through a Program

The E6000 emulator provides a range of options for single stepping through a program, executing an instruction or statement at a time. The alternative step commands listed in table 4.4 are provided.

Table 4.4 Step Commands

Command	Description
Step In	Executes every statement, including statements within functions.
Step Over	Executes function calls in a single step, without stepping through every statement in the called function.
Step Out	Executes out of a function and stops at the next statement in the program that called it.
Step…	Allows you to step repeatedly at a specified rate.

4.8.1 Single Stepping

- Set a breakpoint at PC=H'1036.
- Give one more **Step In** command to execute up to the **sort** function call.

The statement **sort (section1, ID);** will be highlighted.

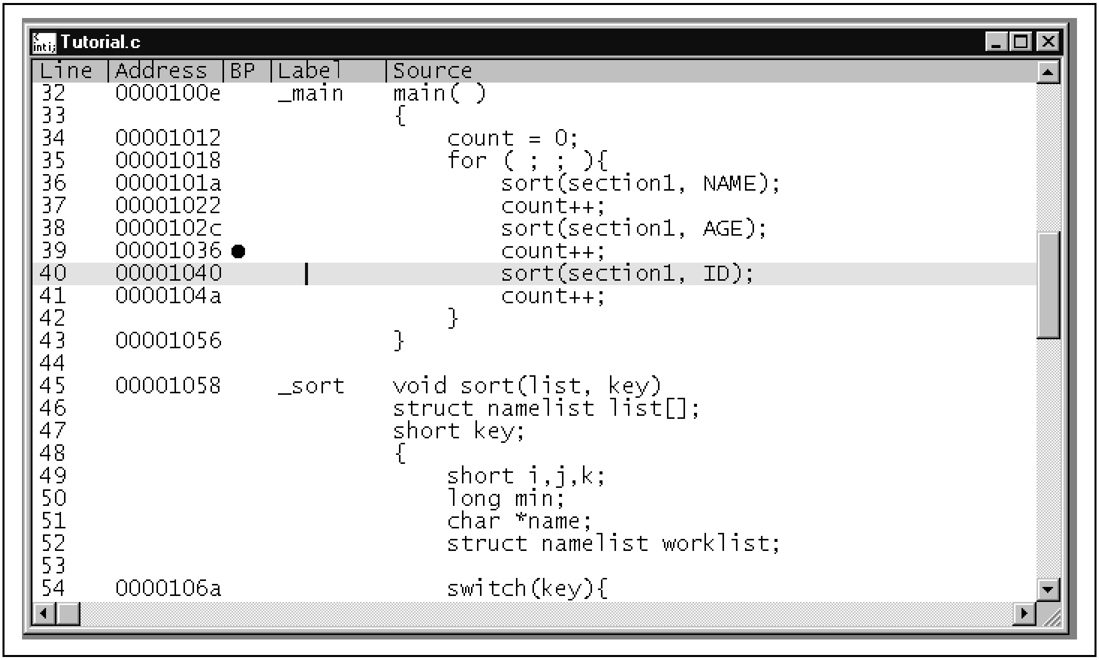


Figure 4.24 Program Window Display after Step In Command Execution (1)

- Now choose **Step In** from the **Run** menu, or click on the **Step In** button in the toolbar.

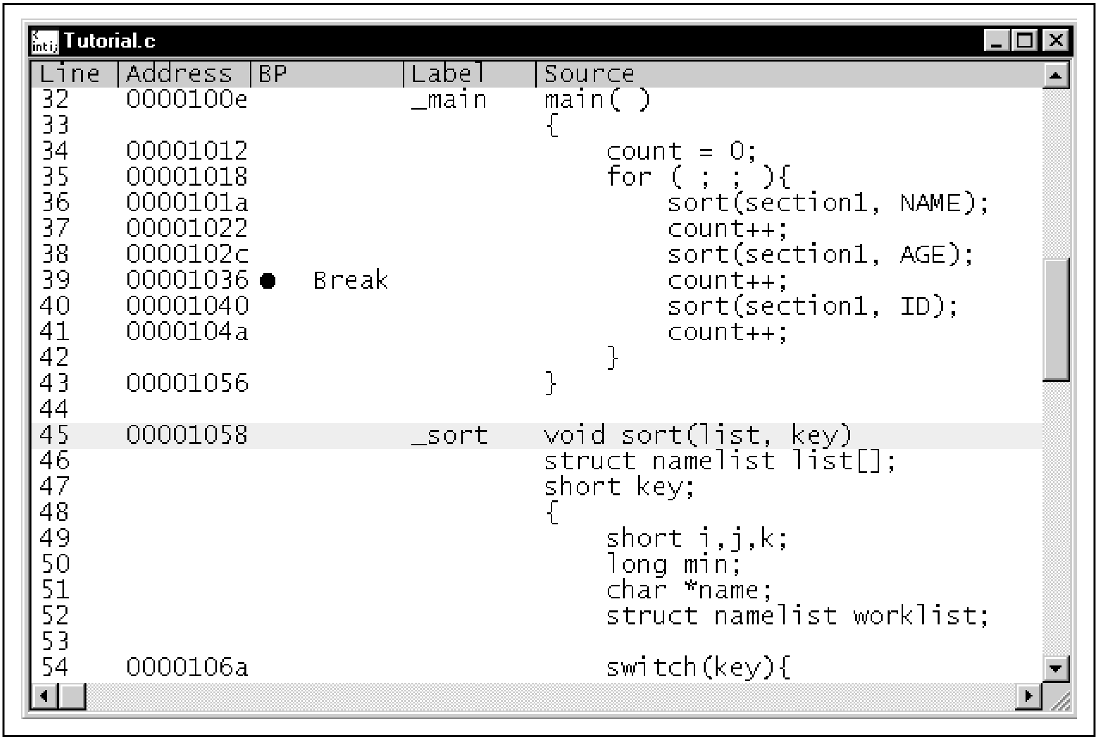


Figure 4.25 Program Window Display after Step In Command Execution (2)

Execute out of the function, and back to the next statement in the main program, by choosing **Step Out** from the **Run** menu, or clicking the **Step Out** button in the toolbar.

Address **H'104a** will be highlighted.

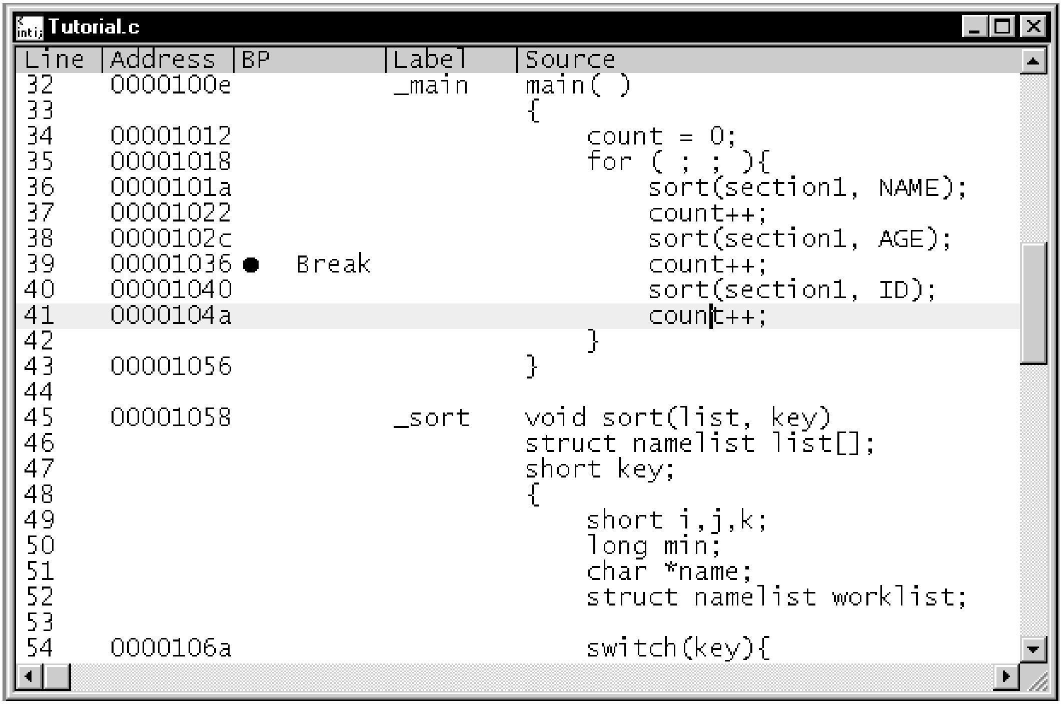


Figure 4.26 Program Window Display after Step Out Command Execution

- Give two more **Step In** commands to execute up to the **sort** function call.

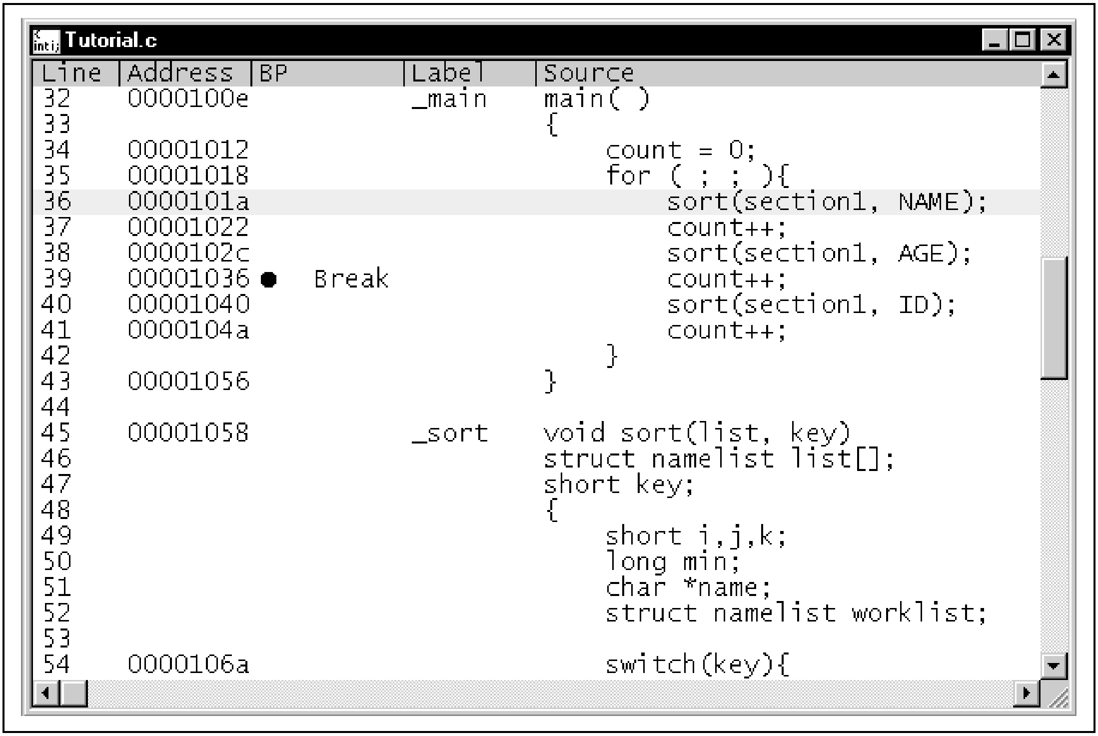


Figure 4.27 Program Window Display after Step In Command Execution (3)

4.8.2 Stepping Over a Function

The **Step Over** command executes a function, without single-stepping through the body of the function, and stops at the next statement in the main program.

- Choose **Step Over** from the **Run** menu, or click the **Step Over** button in the toolbar.

The program executes the **sort** function and stops at address **H'1022**.

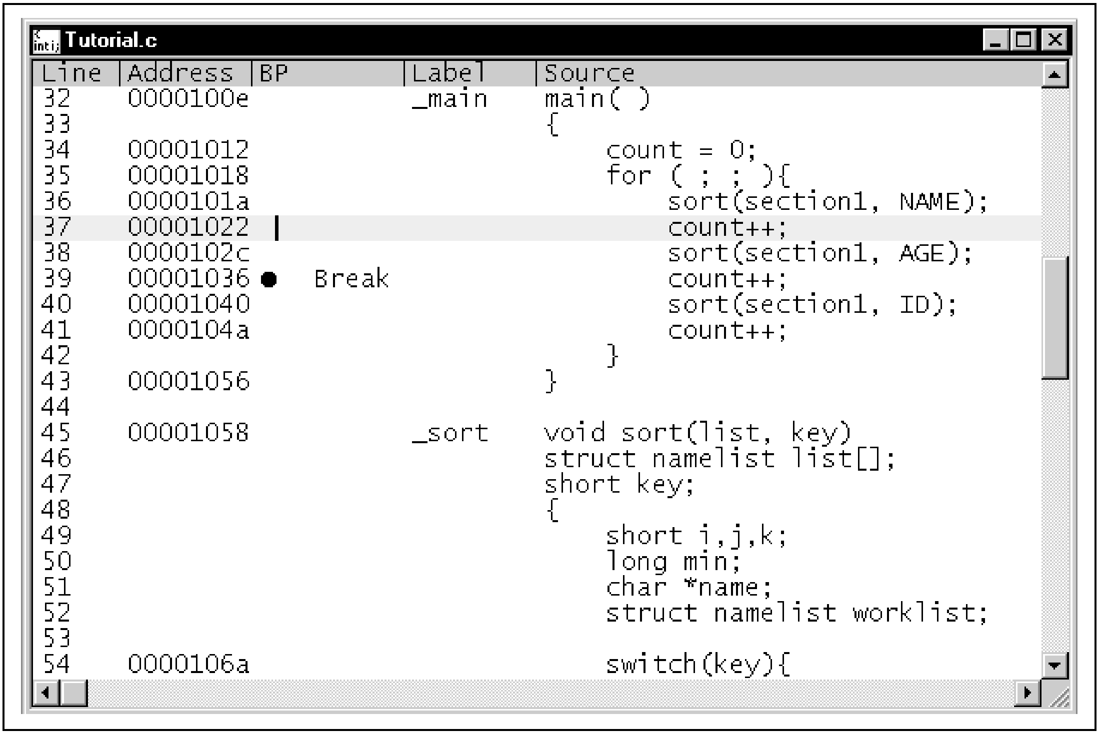


Figure 4.28 Program Window Display after Step Over Command Execution

4.8.3 Watching Local Variables

You can watch local variables in a function using the **Local** window. For example, we will examine the local variables in the function **sort**.

- Choose **Step In** from the **Run** menu to start executing the function **sort**, or click the **Step In** button in the toolbar twice.

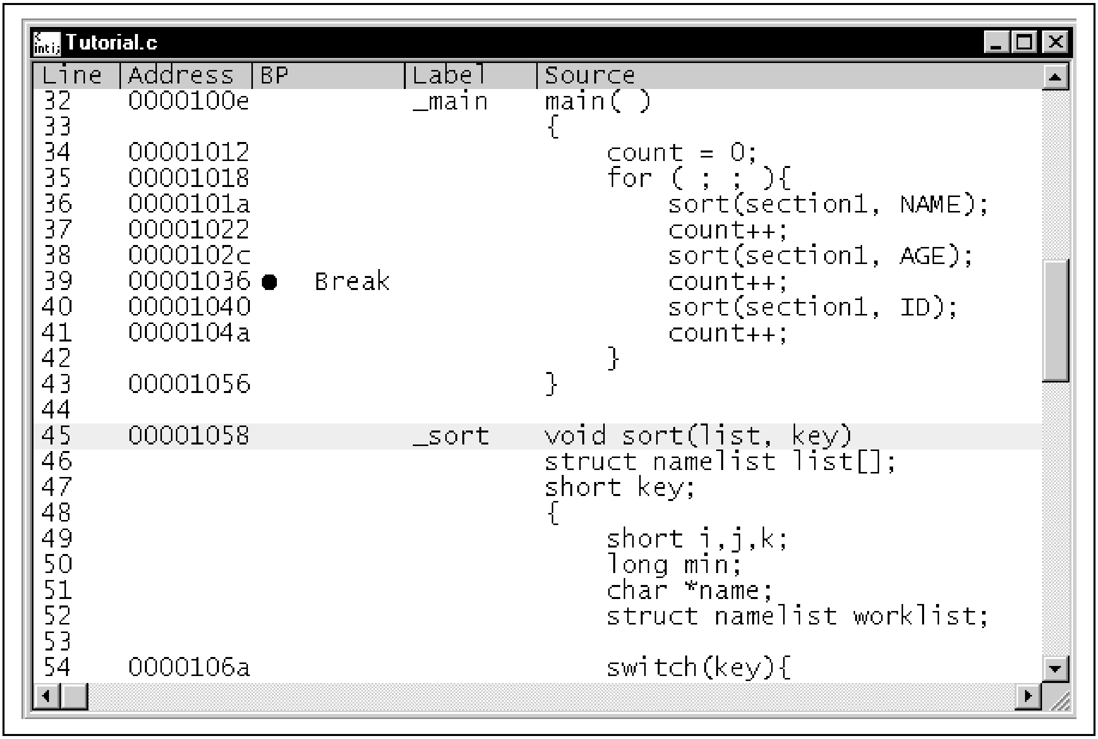


Figure 4.29 Program Window Display after Step In Command Execution (4)

- Open the **Locals** window by choosing **Locals** from the **View** menu or clicking the **Locals** button in the tool bar.

Initially the **Locals** window is empty because the local variables have not yet been declared.

- Choose **Step In** from the **Run** menu eleven times to give ten more steps, or click the **Step In** button in the toolbar eleven times.

The **Locals** window will now show the local variables and their values.

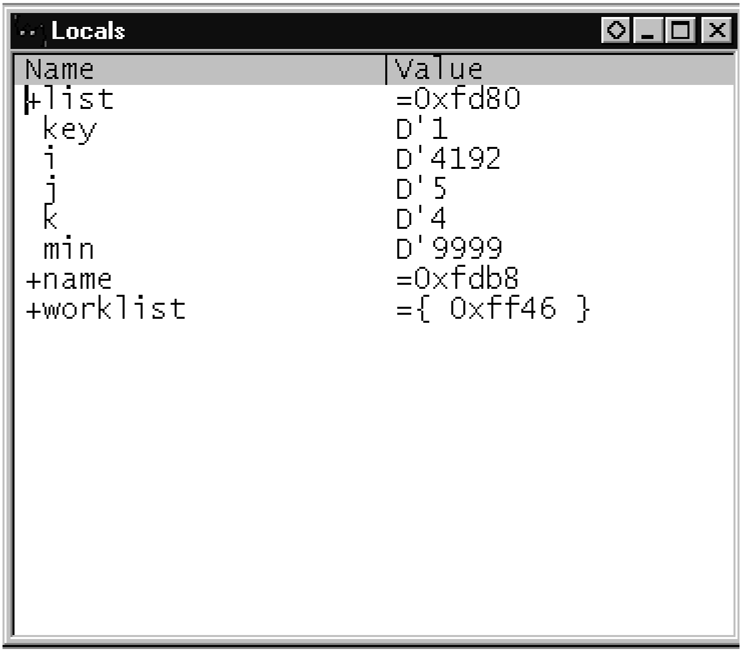


Figure 4.30 Locals Window

- Double-click the **+** symbol in front of the variable **worklist** in the **Locals** window to display the individual elements of the array **worklist**.

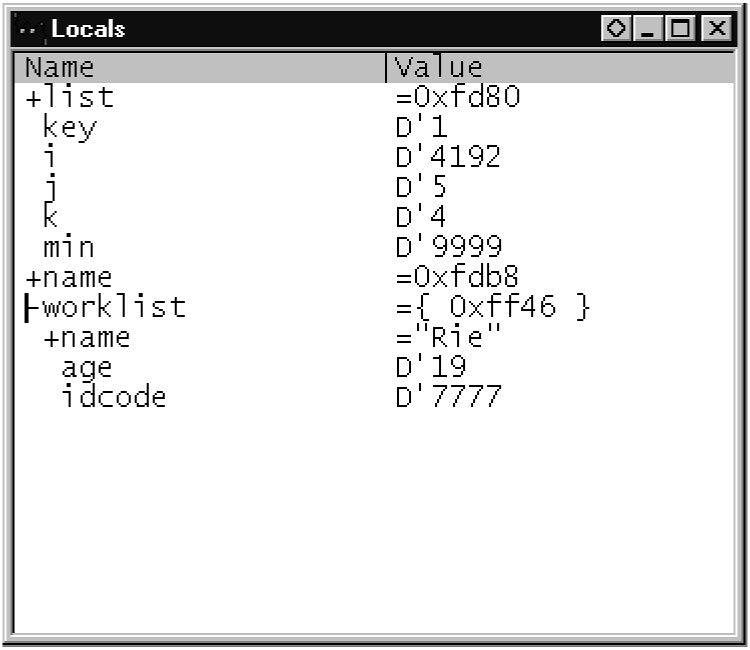


Figure 4.31 Displaying Individual Elements in an Array

- Choose **Step Out** from the **Run** menu to return to the main program, or click the **Step Out** button in the toolbar.

4.9 Using the Complex Event System

So far in this tutorial we have monitored the behavior of the program by observing the contents of an area of memory in the **Memory** window, or the values of variables in the **Watch** window and **Locals** windows.

Sometimes the action of a program is too complex to allow us to do this. For example, how can we know when the program accesses the target address H'109E?

The complex event system allows you to do this.

4.9.1 Defining a Complex Breakpoint

Now define a complex breakpoint to monitor this part as follows:

- Choose **Breakpoint** from the **View** menu to display the **Breakpoints** window, or click the **Breakpoint** button in the toolbar.

- To set a new breakpoint, click the right button of the mouse in the **Breakpoints** window and choose **Add….**

The **Breakpoint/Event Properties** dialog box allows you to define the breakpoint's properties.

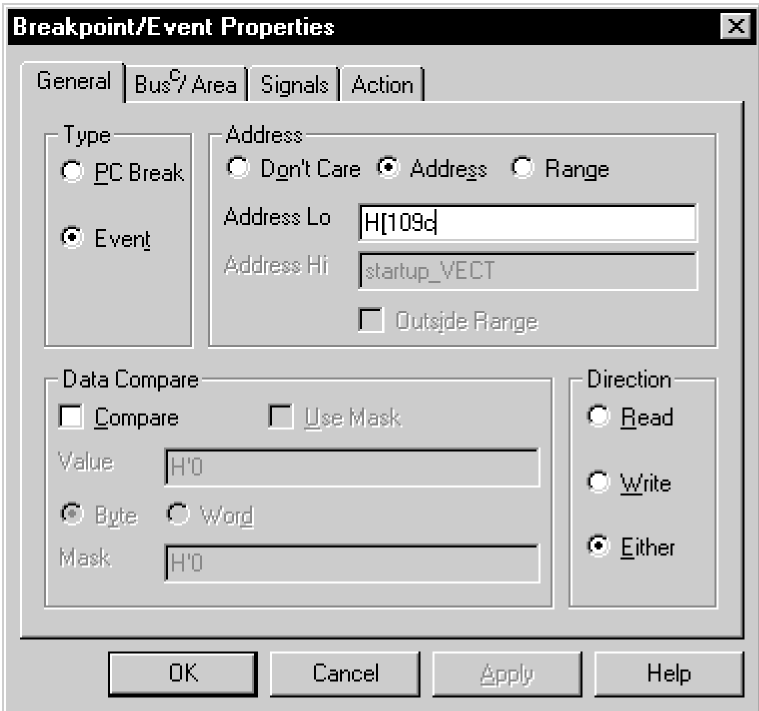


Figure 4.32 Breakpoint/Event Properties Dialog Box

- Set **Type** to **Event** and enter the address **H'109c** into the **Address Lo** box as an event condition.
- Click **OK** to define the breakpoint.

This will cause a break whenever the address H'109c is accessed, either for a read or a write.

The **Breakpoints** window shows the new complex breakpoint you have defined.

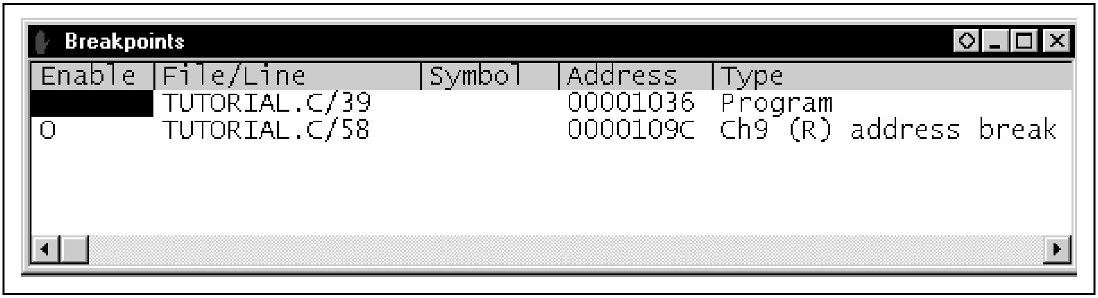


Figure 4.33 Breakpoints Window (After Addition)

- Run the program from the current position, by choosing **Go** from the **Run** menu, or click the **Go** button in the toolbar.

Execution will stop at address **H'109c**.

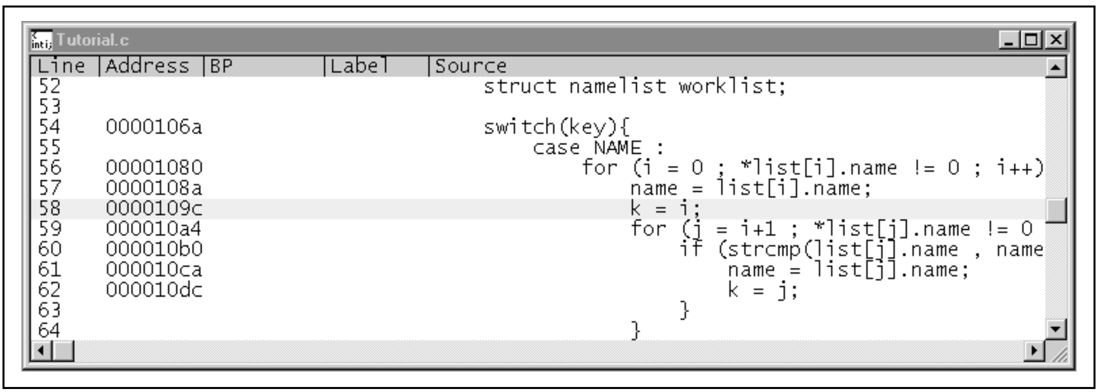


Figure 4.34 Program Break

The status bar will display **BREAK = Event Break** to indicate that the break was caused by event condition satisfaction.

4.10 Using the Trace Buffer

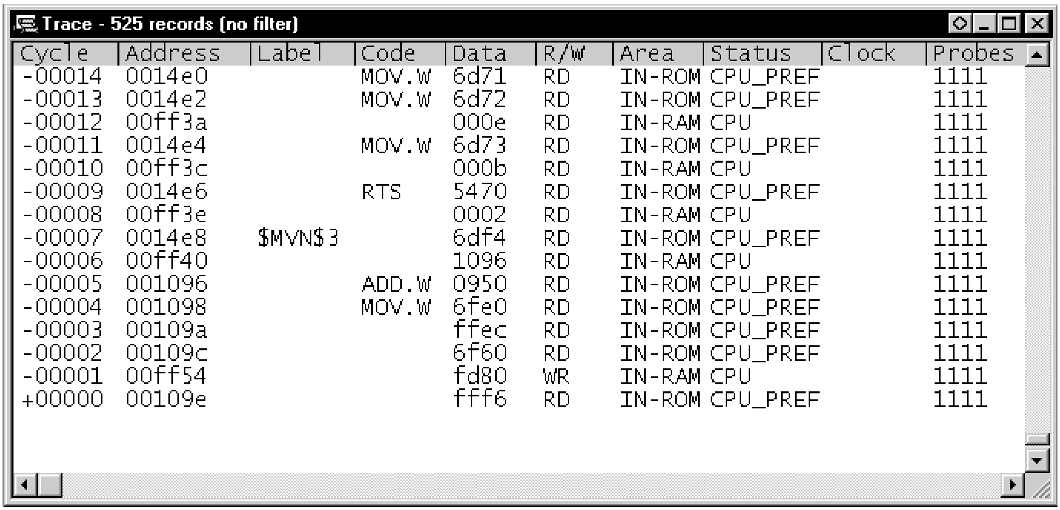
The trace buffer allows us to look back over previous MCU cycles to see exactly what the MCU was doing prior to a specified event.

4.10.1 Displaying the Trace Buffer

Having identified the point in the program where the program accesses, we can use the trace buffer to look back to see what accesses took place.

- Open the **Trace** window by choosing **Trace** from the **View** menu, or click the **Trace** button in the toolbar.

If necessary scroll the window down so that you can see the last few cycles. The **Trace** window is displayed, as shown in figure 4.35.



Trace - 525 records (no filter)

Cycle	Address	Label	Code	Data	R/W	Area	Status	Clock	Probes
-00014	0014e0		MOV.W	6d71	RD	IN-ROM	CPU_PREF		1111
-00013	0014e2		MOV.W	6d72	RD	IN-ROM	CPU_PREF		1111
-00012	00ff3a			000e	RD	IN-RAM	CPU		1111
-00011	0014e4		MOV.W	6d73	RD	IN-ROM	CPU_PREF		1111
-00010	00ff3c			000b	RD	IN-RAM	CPU		1111
-00009	0014e6		RTS	5470	RD	IN-ROM	CPU_PREF		1111
-00008	00ff3e			0002	RD	IN-RAM	CPU		1111
-00007	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111
-00006	00ff40			1096	RD	IN-RAM	CPU		1111
-00005	001096		ADD.W	0950	RD	IN-ROM	CPU_PREF		1111
-00004	001098		MOV.W	6fe0	RD	IN-ROM	CPU_PREF		1111
-00003	00109a			ffec	RD	IN-ROM	CPU_PREF		1111
-00002	00109c			6f60	RD	IN-ROM	CPU_PREF		1111
-00001	00ff54			fd80	WR	IN-RAM	CPU		1111
+00000	00109e			fff6	RD	IN-ROM	CPU_PREF		1111

Figure 4.35 Trace Window

- If necessary adjust the width of each column by dragging the column dividers on either side of the labels just below the title bar.

Note: For the H8/3802 series, the clock count is not displayed. When execution is stopped by a program (PC) break, **Data** is displayed in the **Code** column, and **5770** is displayed in the **Data** column.

4.10.2 Setting a Trace Filter

Currently the **Trace** window shows all the MCU cycles.

- Clear the **Trace** window with the right button of the mouse, and select **Clear** from the pop-up menu to delete the existing trace buffer.
- Similarly, click **Filter…** to display the **Trace Filter** dialog box.

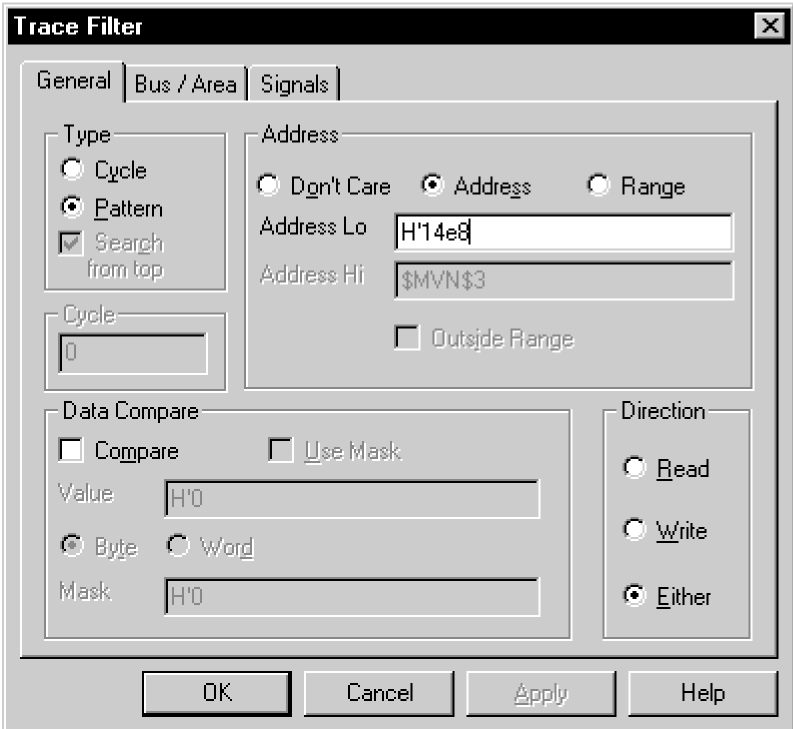


Figure 4.36 General Panel in Trace Filter Dialog Box

This allows you to define a filter to restrict which cycles are displayed in the trace buffer.

- If necessary click **General** to show the **General** panel.
- Choose **Pattern** in the **Type** section.
- In the **Address** section click **Address** and type **H'14e8** in the **Address Lo** field.
- Click **Bus / Area** to display the **Bus / Area** panel.
- Cancel the selection of **Don’t Care** and set **Bus State** to **CPU_PREFETCH**.

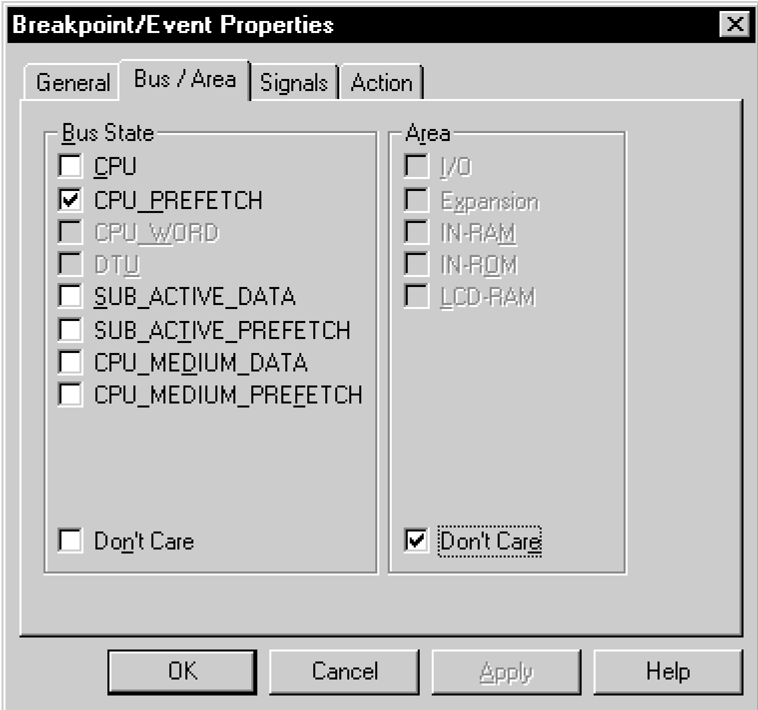
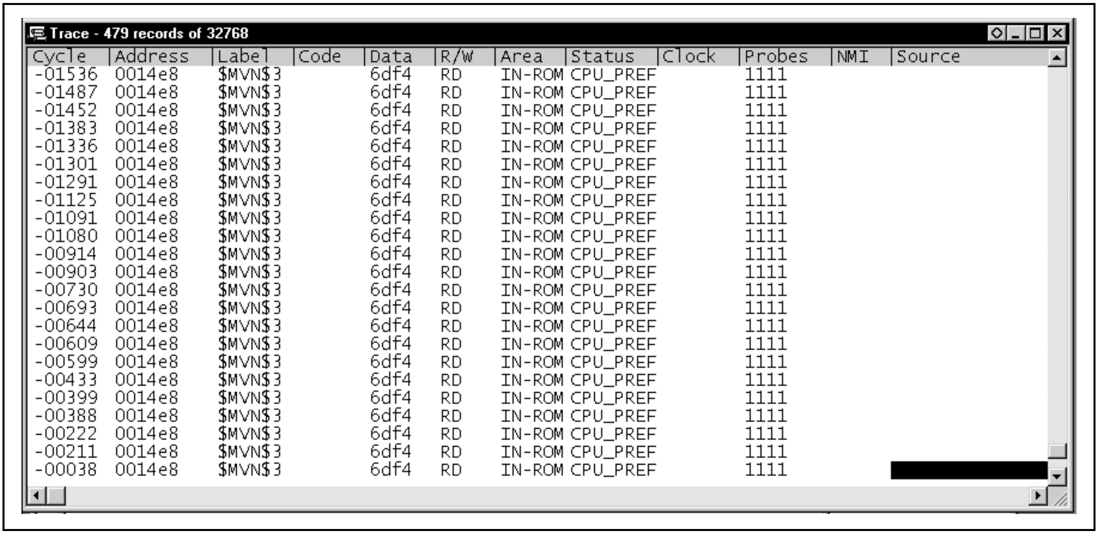


Figure 4.37 Bus / Area Panel in Trace Filter Dialog Box

- Click **OK** to save the trace filter.
- Open the **Breakpoints** window, by choosing **Breakpoints** from the **View** menu and delete the breakpoints you defined earlier.
- Then choose **Go** from the **Run** menu to execute to the end of the program.
- Choose **Halt** from the **Run** menu to halt execution so that you can view the trace buffer.

The **Trace** window will show the cycle in which MCU accesses H'14e8, as shown in figure 4.38.



Trace - 479 records of 32768

Cycle	Address	Label	Code	Data	R/W	Area	Status	Clock	Probes	NMI	Source
-01536	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01487	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01452	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01383	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01336	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01301	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01291	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01125	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01091	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-01080	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00914	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00903	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00730	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00693	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00644	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00609	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00599	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00433	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00399	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00388	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00222	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00211	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		
-00038	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111		

Figure 4.38 Showing Trace Buffer Contents

4.11 Save the Session

Before exiting it is good practice to save your session, so that you can resume with the same E6000 emulator and HDI configuration at your next debugging session.

- Choose **Save Session** from the **File** menu.
- Choose **Exit** from the **File** menu to terminate the HDI.

4.11.1 What Next?

This tutorial has introduced you to some of the key features of the E6000 emulator, and their use in conjunction with the HDI. By combining the emulation tools provided in the E6000 emulator you can perform extremely sophisticated debugging, allowing you to track down hardware and software problems very efficiently by precisely isolating and identifying the conditions under which they occur.

For details on HDI usage, refer to Hitachi Debugging Interface User’s Manual.

Section 5 Reference

This section gives reference information about the features of the HDI specific to the H8/3802 series of microcomputers. For information about the general features of the HDI, common to all targets, refer to the Hitachi Debugging Interface User's Manual, supplied separately.

Table 5.1 shows the correspondence between the HDI menus and the descriptions in Hitachi Debugging Interface User's Manual (HDI manual) and this manual.

Table 5.1 Correspondence Between HDI Menus and Descriptions in Manuals

Menu Bar	Pull-Down Menu	HDI Manual	This Manual
File Menu	New Session...	O	—
	Load Session...	O	—
	Save Session	O	4.11
	Save Session As...	O	—
	Load Program...	O	4.5
	Initialize	O	—
	Exit	O	—
Edit Menu	Cut	O	—
	Copy	O	—
	Paste	O	—
	Find...	O	—
	Evaluate...	O	—
View Menu	Breakpoints	O	4.6.4, 4.9.1, 5.2, 5.3
	Command Line	O	5.7
	Disassembly...	O	—
	I/O Area	O	—
	Labels	O	—
	Locals	O	4.8.3
	Memory...	O	4.7.1
	Performance Analysis	O	—
	Registers	O	4.6.3
	Source...	O	4.5

Notes: 1. O : Described
— : Not described
2. The numbers in the This Manual columns are the reference section numbers.

Table 5.1 Correspondence Between HDI Menus and Descriptions in Manuals (cont)

Menu Bar	Pull-Down Menu	HDI Manual	This Manual
View Menu (cont)	Status	O	4.6.2
	Trace	O	4.10, 5.5
	Watch	O	4.6
Run Menu	Reset CPU	O	—
	Go	O	4.6.2
	Reset Go	O	4.6.2
	Go To Cursor	O	—
	Set PC To Cursor	O	—
	Run...	O	—
	Step In	O	4.8
	Step Over	O	4.8
	Step Out	O	4.8
	Step...	O	—
	Halt	O	—
Memory Menu	Refresh	O	—
	Load...	O	—
	Save...	O	—
	Verify...	O	—
	Test...	O	—
	Fill...	O	—
	Copy...	O	—
	Compare...	O	—
	Configure Map...	O	4.4.2
	Configure Overlay...	O	—
Setup Menu	Status bar	O	—
	Options...	O	—
	Radix	O	—
	Customize	O	—
	Configure Platform...	O	4.4.1, 5.1

Notes: 1. O : Described

— : Not described

2. The numbers in the This Manual columns are the reference section numbers.

Table 5.1 Correspondence Between HDI Menus and Descriptions in Manuals (cont)

Menu Bar	Pull-Down Menu	HDI Manual	This Manual
Window Menu	Cascade	O	—
	Tile	O	—
	Arrange Icons	O	—
	Close All	O	—
Help Menu	Index	O	—
	Using Help	O	—
	Search for Help on	O	—
	About HDI	O	—

Notes: 1. O : Described
— : Not described
2. The numbers in the This Manual columns are the reference section numbers.

5.1 Configuration Dialog Box

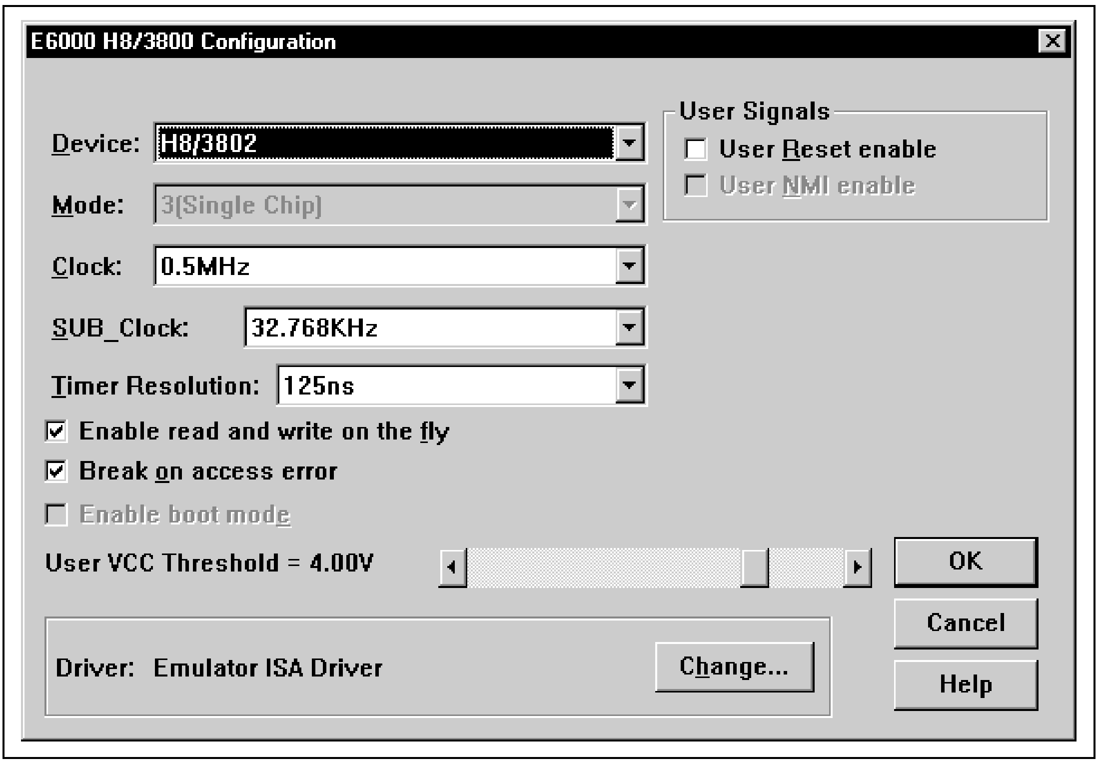


Figure 5.1 Configuration Dialog Box

The **Configuration** dialog box allows you to set up the E6000 emulator.

To display the configuration dialog box, choose **Configure Platform…** from the **Setup** menu.

Table 5.2 explains the options provided in the **Configuration** dialog box.

Table 5.2 Configuration Options

Option	Description
Device	Specifies the MCU device.
Mode	Specifies the MCU operating mode. Mode is fixed to 3.
Clock	Specifies the MCU clock rate. Can be set to: 0.5MHz, 2 MHz, 8 MHz, or Target /2 (H8/3802 series system clock). Specifies the MCU subclock rate. Can be set to 32.768 kHz, 38.4 kHz, 307.2 kHz, or Target.
Timer Resolution	Specifies the minimum time used for performing execution time measurements. Can be set to one of the following values: 20 ns, 125 ns, 250 ns, 500 ns, 1 µs, 2 µs, 4 µs, 8 µs, or 16 µs.
User Signals	Allows you to disable or enable the user reset signal. When the box is checked the signal is enabled.
Enable read and write on the fly	Allows HDI access to user memory in run mode.
Enable boot mode	Allows boot programming operation for flash memory in the MCU. This option cannot be used in the H8/3802 series.
Break on access error	Causes all illegal accesses to halt emulation. If not checked, all writes to ROM or accesses to the guarded area are ignored.
User VCC Threshold	Monitors the user’s system voltage level and, if it falls below the value set by the threshold, informs the user that the User VCC is down using the **System Status** window.

5.2 Breakpoints

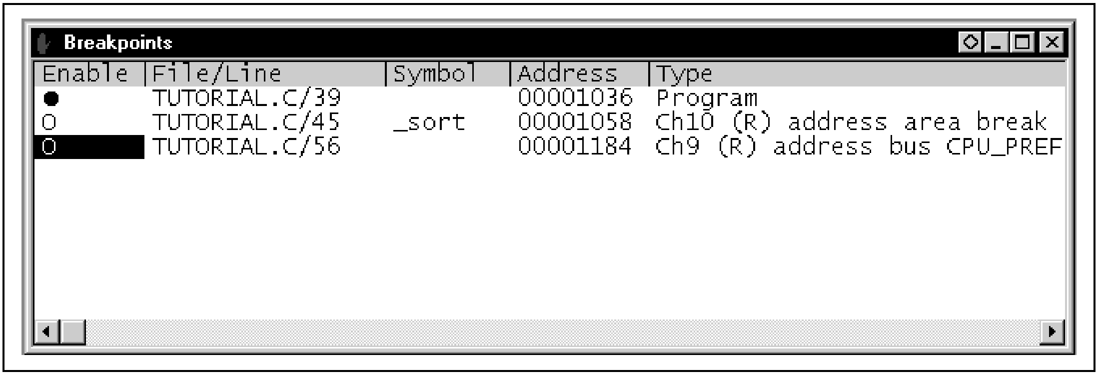


Figure 5.2 Breakpoints Window

The **Breakpoints** window displays a list of all the breakpoints that have been defined.

To display the **Breakpoints** window choose **Breakpoints** from the **View** menu.

To edit an existing breakpoint double-click it, or select it in the **Breakpoints** list and choose **Edit...** from the pop-up menu.

To enable or disable a breakpoint select it in the **Breakpoints** list and choose **Disable/Enable** from the pop-up menu. When a breakpoint is enabled • is shown in the **Enable** column.

To delete a breakpoint select it in the breakpoint list and choose **Delete** from the pop-up menu, or **Delete All** to delete all the breakpoints.

To define a new breakpoint choose **Add…** from the pop-up menu to display the **Breakpoint/Event Properties** dialog box, and define the characteristics of the breakpoint you want to add.

For more information about the **Breakpoint/Event Properties** dialog box see section 5.3, Complex Event System.

5.2.1 Defining Program Breakpoints

To define a program breakpoint set **Type** to **PC Break** and enter the address of the breakpoint in the **Address Lo** field:

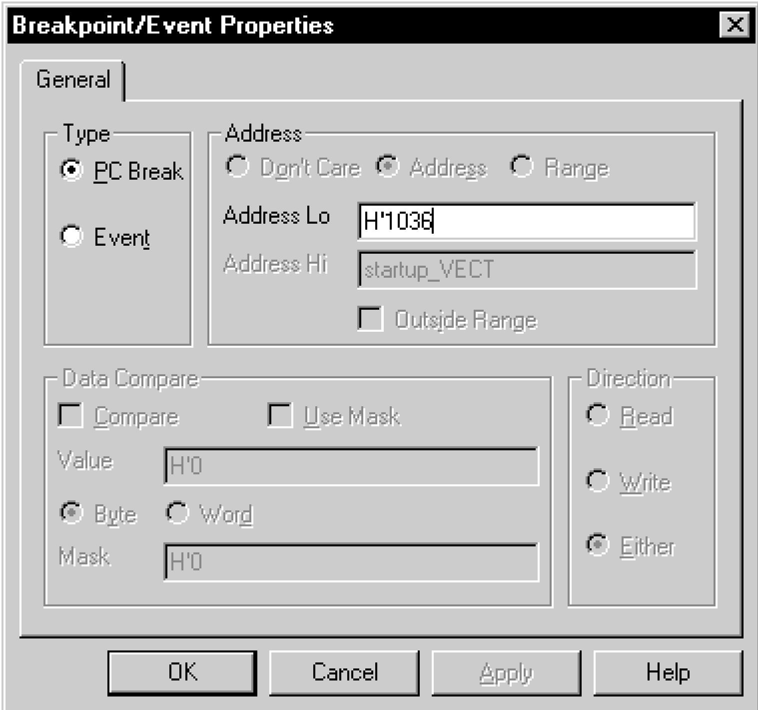


Figure 5.3 Breakpoint/Event Properties Dialog Box

Alternatively, double-click in the **Break** column in the program window.

5.3 Complex Event System

The complex event system (CES) allows you to define events which depend on the state of a specified combination of the MCU signals and provides a unified way of controlling the trace, break, and timing functions of the E6000 emulator.

The complex event system uses the event and range channels to allow you to detect when a specified event has occurred. Up to eight events can be combined into a sequence, in which each event is either activated or deactivated by the occurrence of the previous event in the sequence.

Table 5.3 shows the options that can be specified for event and range channels.

Table 5.3 Event and Range Channel Options

Option	Event	Range
Access to a specified address or within a specified address range.	O	O
Access outside a specified address range.	O	
A specific value of data, with an optional mask.	O	O
A specific MCU access direction (read or write).	O	O
A specific MCU access type (instruction prefetch, etc).	O	O
A specific MCU access area (internal ROM, RAM, etc).	O	O
A signal state on one or more of the four external probes.	O	O
A specified number of times that the event must be triggered.	O	
Can be combined into a sequence.	O	

O: Can be specified.

The **Breakpoint/Event Properties** dialog box allows you to define complex events for use with breakpoints, trace, and execution timing.

To define an event breakpoint set **Type to Event**. The **Breakpoint/Event Properties** dialog box then provides four panels of options to allow you define all the characteristics of the event used by the breakpoint: **General**, **Bus / Area**, **Signals**, and **Action**:

5.3.1 General

The **General** properties panel allows you to define the address and data access characteristics of the event channel.

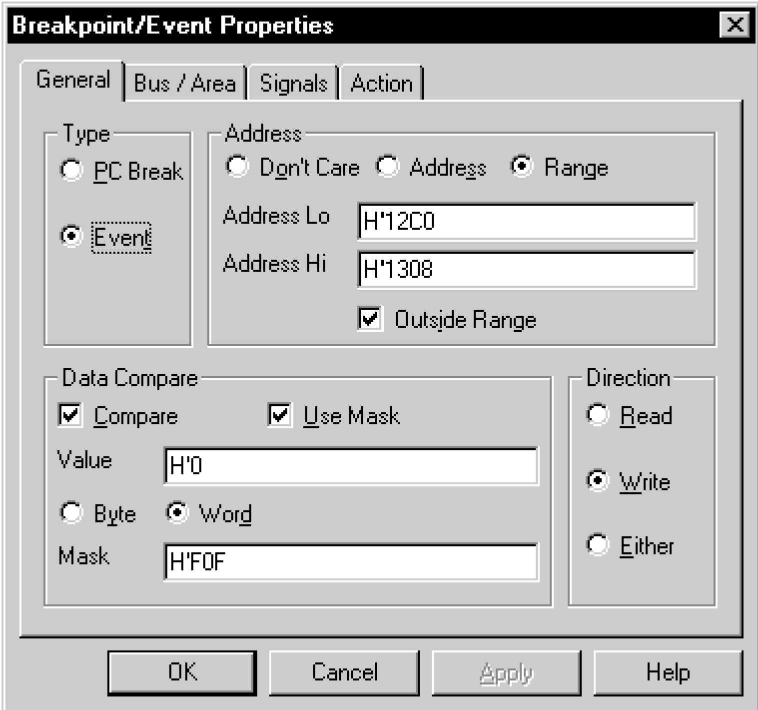


Figure 5.4 General Panel

Address: Allows you to activate the channel when an address, or a range of addresses is accessed. Select **Outside Range** to specify that accesses to addresses outside the specified range should trigger the channel.

Data Compare: Allows you to trigger the channel on a specific data value. Select **Use Mask** to specify a mask which will be ANDed with the data before comparing it with the value.

Direction: Allows you to specify either read, write, or read and write accesses to trigger the channel.

5.3.2 Bus / Area

Allows you to trigger the channel on specific bus states or memory areas accessed.

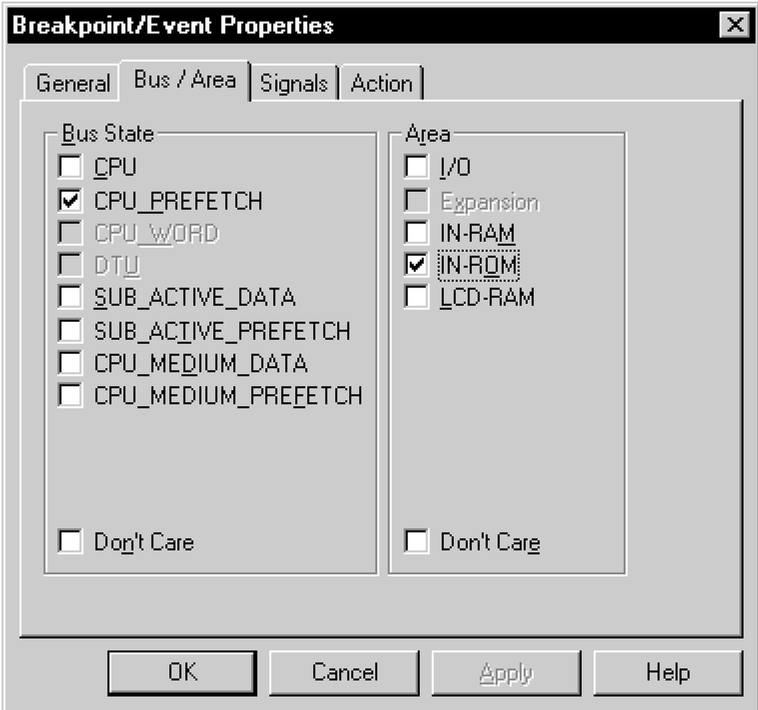


Figure 5.5 Bus / Area Panel

5.3.3 Signals

Specifies that the event should be triggered on a specific combination of the four external probe signals.

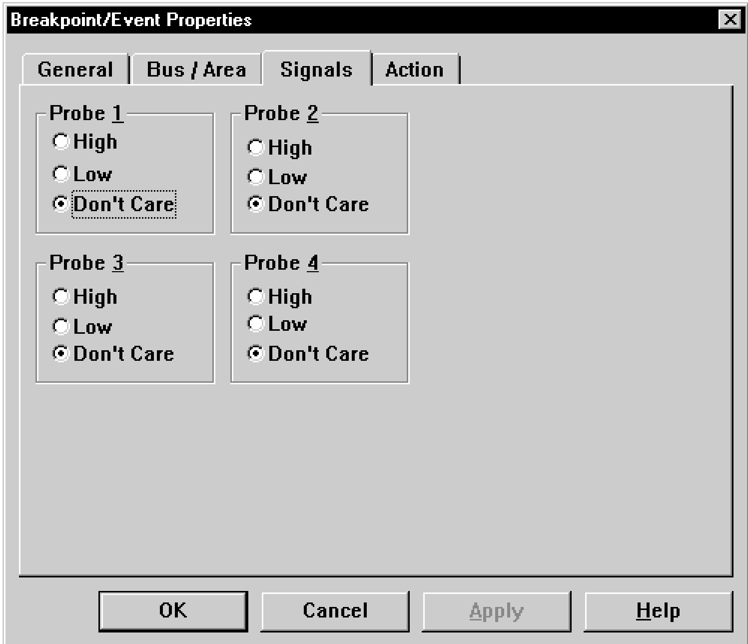


Figure 5.6 Signals Panel

5.3.4 Action

Specifies the action when the event is triggered.

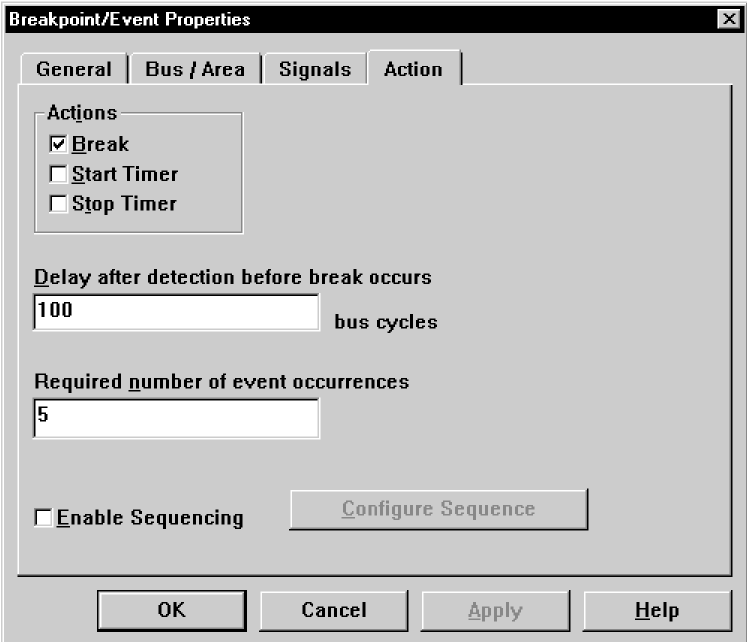


Figure 5.7 Action Panel

Table 5.4 lists the actions that can be specified.

Table 5.4 Specifiable Actions

Action	Description
Break	Halts program execution.
Start Timer	Starts the execution timer; see section 5.1, Configuration Dialog Box, for more information about the timer resolution.
Stop Timer	Stops the execution timer.

To delay activation of the channel for a specified number of bus cycles after it is triggered, enter the number of bus cycles in the **Delay after detection before break occurs** field.

To delay activation of the channel until it has been triggered a specified number of times, enter the required number of event occurrences in the **Required number of event occurrences** field.

To create a sequence of events select the **Enable Sequencing** option for all events that are going to form part of the sequence.

5.3.5 Event Sequencing

To configure the sequence check **Enable Sequencing** and click the **Configure Sequence** button in the **Action** panel in any of the breakpoints. The **Event Sequencing** dialog box will be displayed.

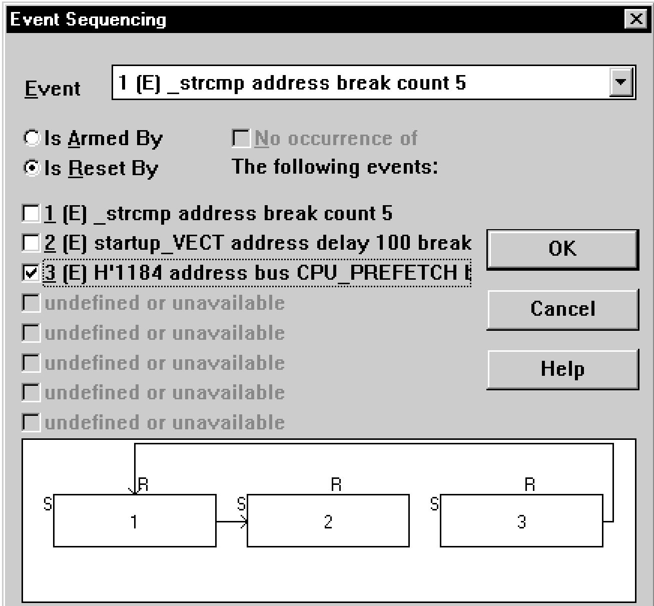


Figure 5.8 Event Sequencing Dialog Box

For each event in the sequence this dialog box allows you to specify one or more other events that will arm it or reset it.

Select the event that you want to configure from the **Event** drop down list box. This gives you a choice of any events for which enable sequencing has been specified.

Then, for the currently selected event, click **Is Armed By** and check the events that should arm the event.

Likewise, click **Is Reset By** and check the events that should reset the event.

5.3.6 Arming Events

For example, to define an event sequence that is triggered only when a sequence of four address reads have occurred you would define:

- 4 is armed by 3.
- 3 is armed by 2.
- 2 is armed by 1.

The **Event Sequencing** dialog box displays a diagrammatic representation of the sequence you have defined.

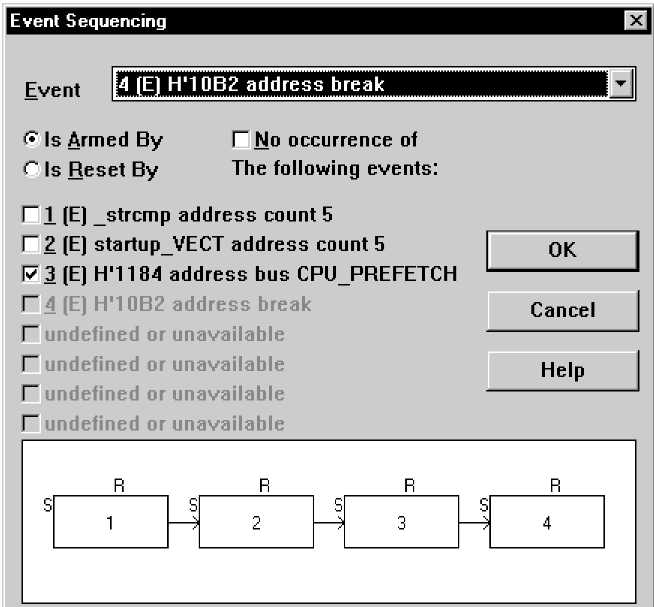


Figure 5.9 Event Sequence Diagram

Note that when defining a sequence only the last event in the sequence should be defined as a break.

5.3.7 Resetting Events

You can also specify that events are reset by another event in the sequence. For example to cause a break if event 2 is followed by event 3 and then by event 4, provided that event 1 has not occurred in the meantime, define the event sequence as follows:

- 4 is armed by 3 and reset by 1.
- 3 is armed by 2 and reset by 1.
- 2 is reset by 1.
- 1 is reset by1.

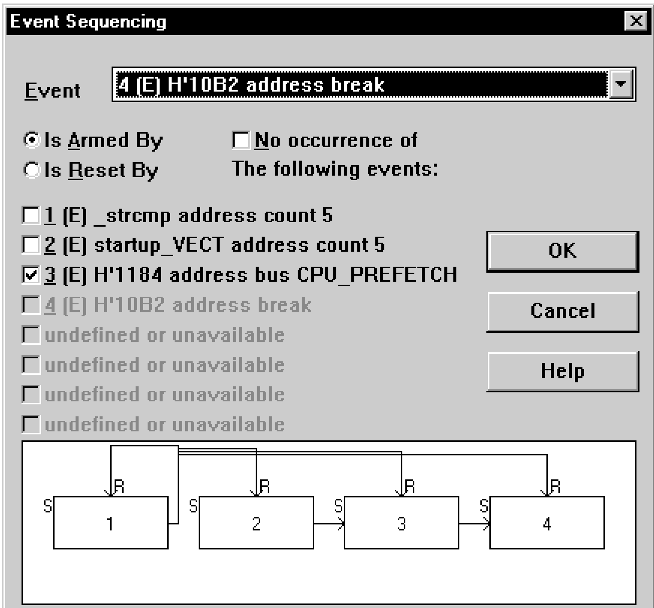


Figure 5.10 Resetting Events

5.4 Memory Mapping Dialog Box

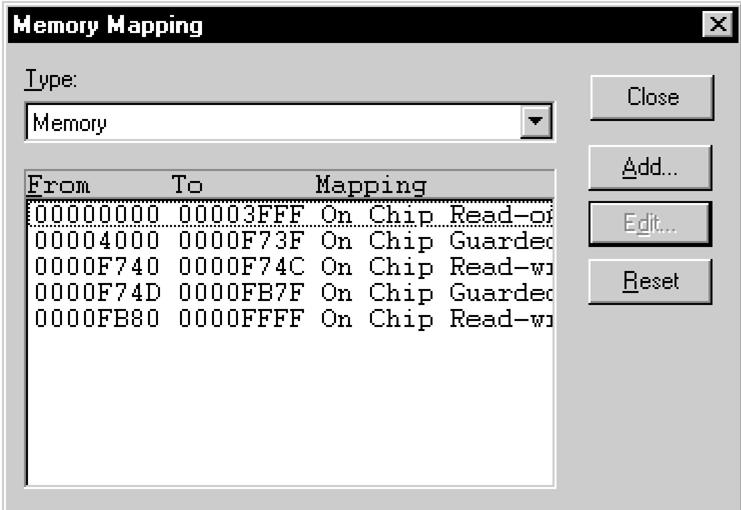


Figure 5.11 Memory Mapping Dialog Box

The **Memory Mapping** dialog box shows the E6000 emulator memory mapping, and allows you to edit it.

To display this dialog box choose **Configure Map...** from the **Memory** menu.

To edit a block of memory double-click it, or select it in the memory mapping list and click **Edit**. The **Edit Memory Mapping** dialog box shows the current setting for the block of memory.

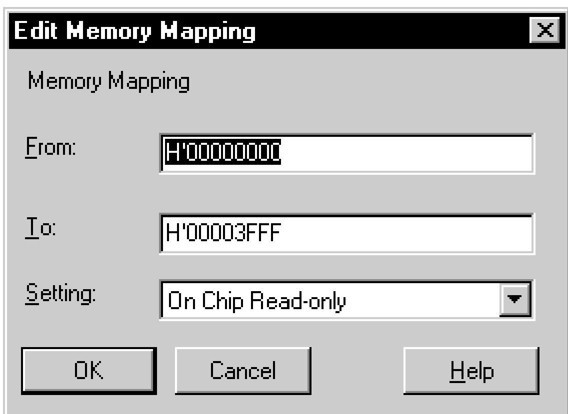


Figure 5.12 Edit Memory Mapping Dialog Box

Specify the range of addresses for the block of memory in the **From** and **To** fields, and select the type of memory from the **Setting** drop down list box. The options listed in table 5.5 are available:

Table 5.5 Memory Type

Memory	Description
Internal	Accesses the MCU internal memory (ROM/RAM).
Emulator	Accesses the emulation memory.

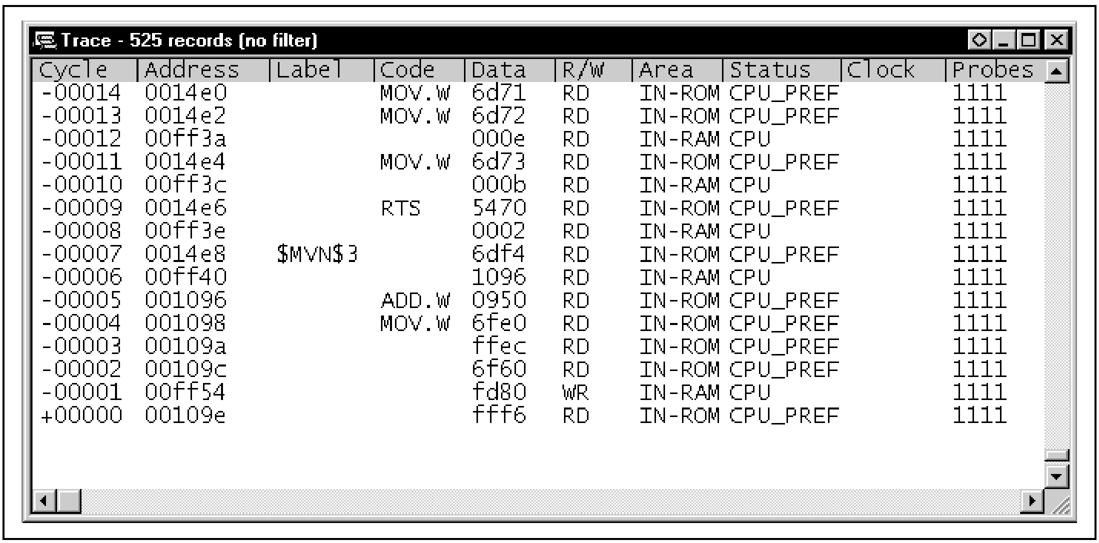
For each of these options you can specify one of the three access types listed in table 5.6:

Table 5.6 Access Types

Access Type	Description
Read-write	RAM.
Read-only	ROM.
Guarded	No access allowed.

Click **Reset** in the **Memory Mapping** dialog box to reset the memory mapping to the default mapping for these selected MCU type and mode.

5.5 Trace Window


Trace - 525 records (no filter)

Cycle	Address	Label	Code	Data	R/W	Area	Status	Clock	Probes
-00014	0014e0		MOV.W	6d71	RD	IN-ROM	CPU_PREF		1111
-00013	0014e2		MOV.W	6d72	RD	IN-ROM	CPU_PREF		1111
-00012	00ff3a			000e	RD	IN-RAM	CPU		1111
-00011	0014e4		MOV.W	6d73	RD	IN-ROM	CPU_PREF		1111
-00010	00ff3c			000b	RD	IN-RAM	CPU		1111
-00009	0014e6		RTS	5470	RD	IN-ROM	CPU_PREF		1111
-00008	00ff3e			0002	RD	IN-RAM	CPU		1111
-00007	0014e8	MVN3		6df4	RD	IN-ROM	CPU_PREF		1111
-00006	00ff40			1096	RD	IN-RAM	CPU		1111
-00005	001096		ADD.W	0950	RD	IN-ROM	CPU_PREF		1111
-00004	001098		MOV.W	6fe0	RD	IN-ROM	CPU_PREF		1111
-00003	00109a			ffec	RD	IN-ROM	CPU_PREF		1111
-00002	00109c			6f60	RD	IN-ROM	CPU_PREF		1111
-00001	00ff54			fd80	WR	IN-RAM	CPU		1111
+00000	00109e			fff6	RD	IN-ROM	CPU_PREF		1111

Figure 5.13 Trace Window

The **Trace** window displays the contents of the trace buffer.

To display the **Trace** window choose **Trace** from the **View** menu.

The data stored in the trace buffer is displayed in both source program and assembly languages for ease of debugging. However, if trace filtering is used then only assembly language is displayed.

Nothing is displayed when this emulator is used.

The values of four external probes are displayed in the **Probes** column. The left-most value indicates probe 4 and the right-most value indicates probe 1. The value 1 shows the high level and 0 shows the low level.

Nothing is displayed in the **NMI** column when this emulator is used.

Click **Clear** to clear the trace buffer, or click **Save** to save the contents of the trace buffer to a file.

By default the trace buffer captures all the execution cycles and retains the last 32768 cycles. You can set up a filter which will restrict the traces displayed from the buffer to specified cycle patterns.

5.5.1 Filter

To define a filter, choose **Filter...** from the pop-up menu in the **Trace** window.

5.5.2 Find

To search for a specific trace in the trace buffer choose **Find…** from the pop-up menu. The same dialog box appears to specify the traces you want to find.

5.5.3 Cycle

To specify a specific cycle as a filter set **Type** to **Cycle** and enter the cycle number in the **Cycle** box.

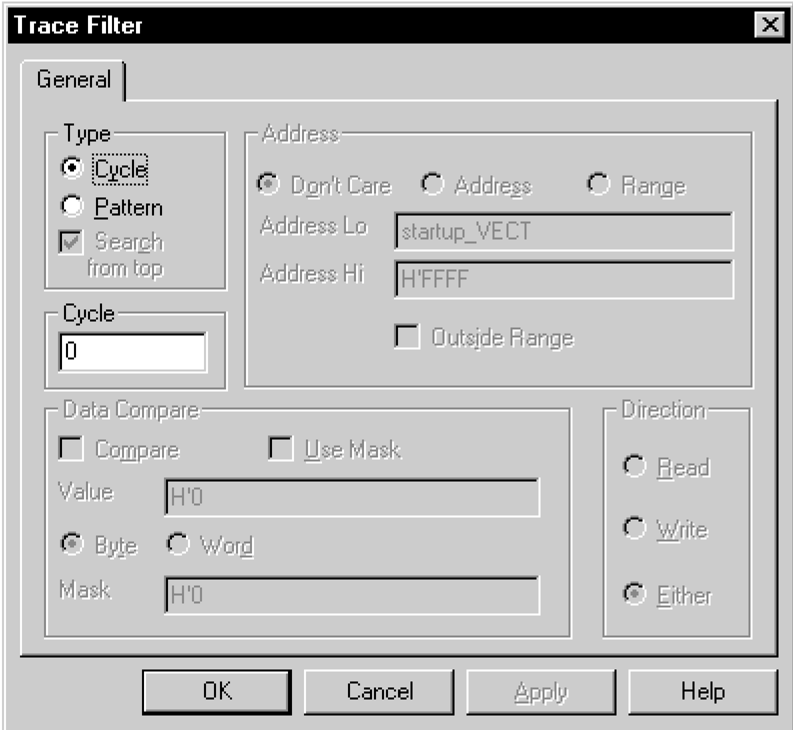


Figure 5.14 Trace Filter Dialog Box

5.5.4 Pattern

To enter a filter pattern set **Type** to **Pattern** and specify the values as required.

The **Trace Filter** dialog box then provides three panels of options to allow you to specify which cycles should be captured: **General**, **Bus / Area**, and **Signals**, see the following.

5.5.5 General

The **General** panel allows you to define the address and data access characteristics of the cycles to be displayed.

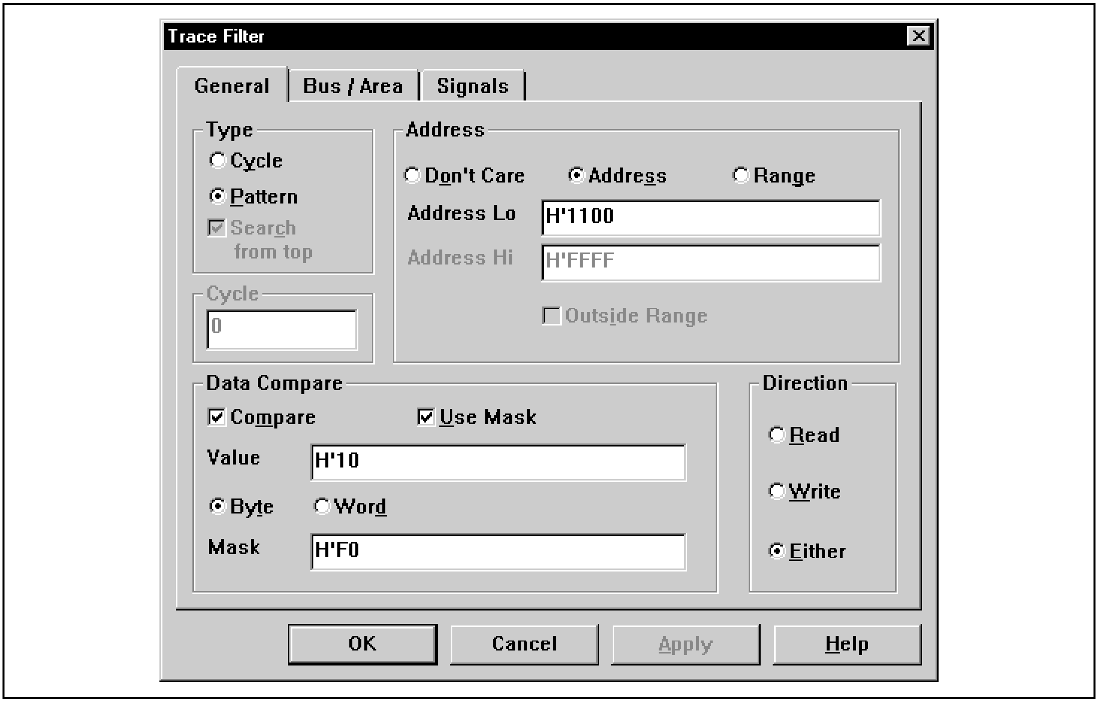


Figure 5.15 General Panel

5.5.6 Bus / Area

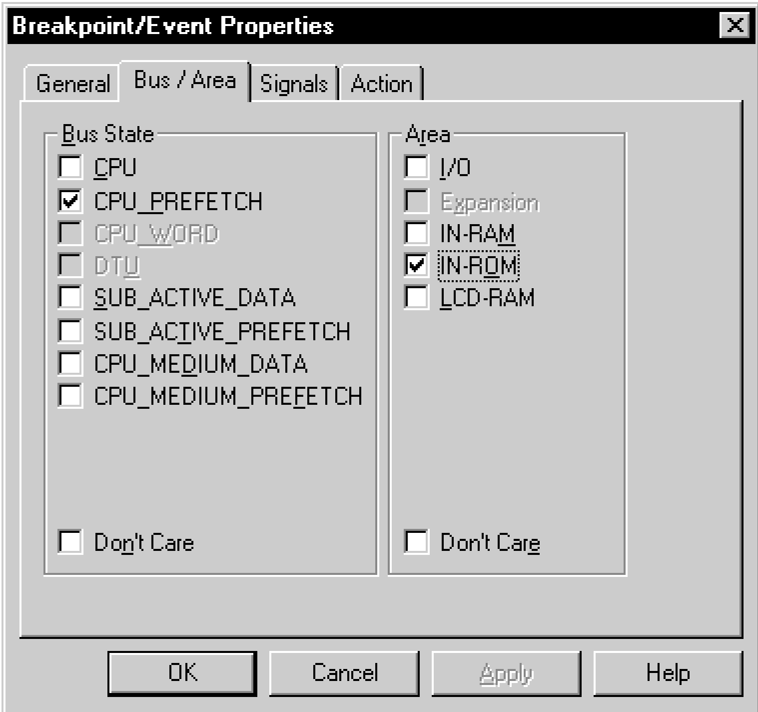


Figure 5.16 Bus / Area Panel

5.5.7 Signals

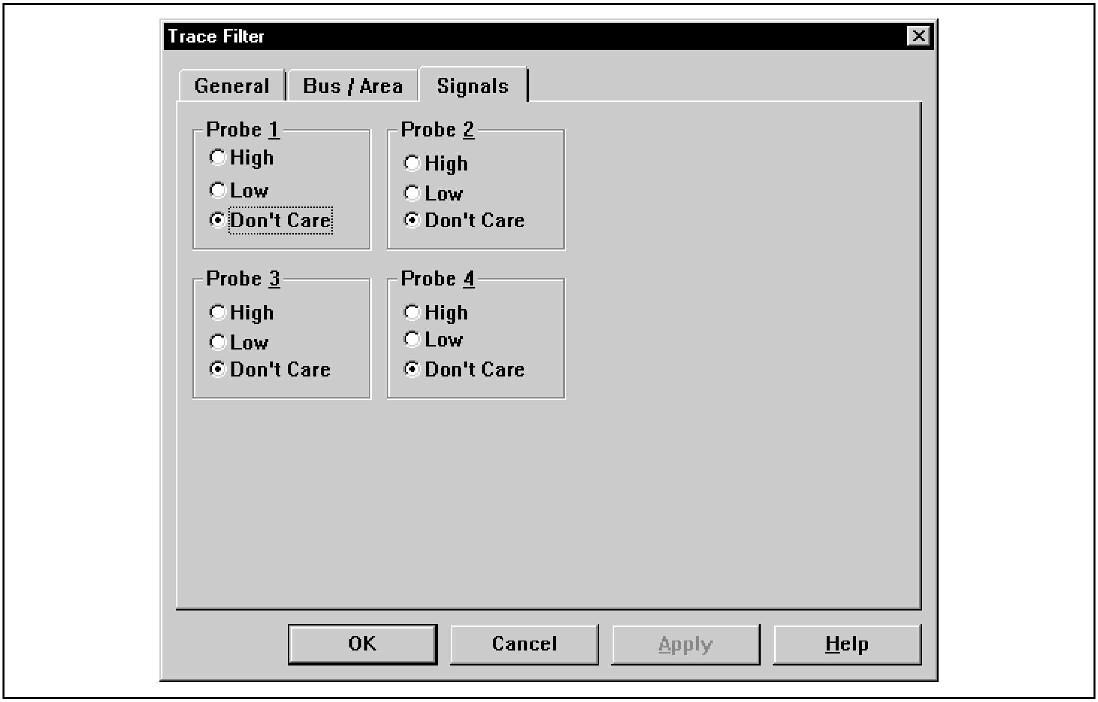


Figure 5.17 Signals Panel

5.6 Trace Acquisition

The buffer can be set up to store all bus cycles or just selected cycles. This is called trace acquisition. To specify the trace acquisition click **Acquisition** in the **Trace** window.

The **Trace Acquisition** dialog box provides the following panels to allow you to specify when trace acquisitions begins.

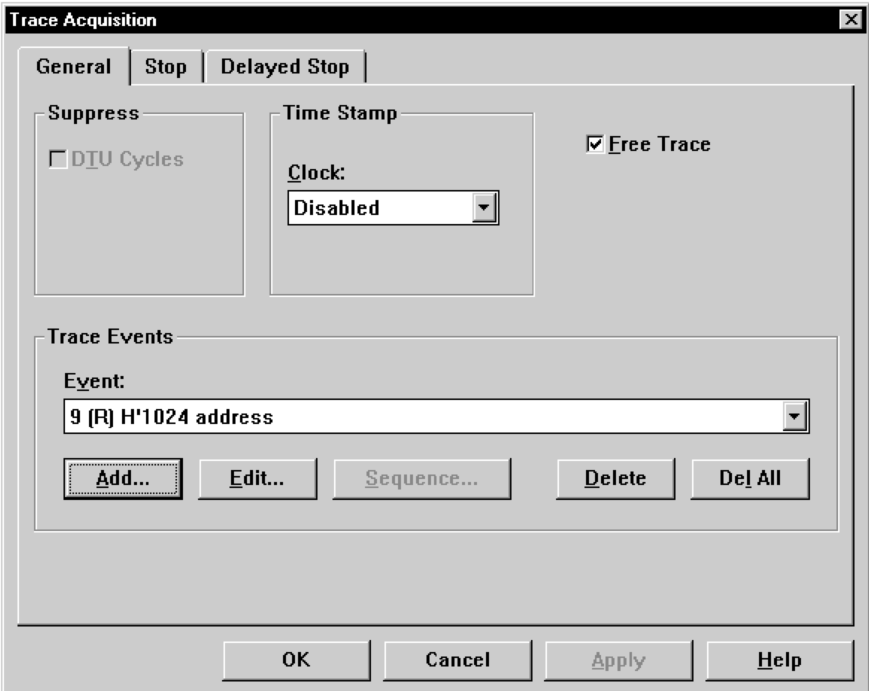


Figure 5.18 General Panel

5.6.1 General

DTU Cycles check box in **Suppress** space is disabled. **Time Stamp** section allows you to acquire the program execution time in the trace buffer. When **Time Stamp** is used, the following information cannot be acquired:

- Area
- Status
- Probes

Also multiplication/division instruction cannot be displayed.

Check the **Free Trace** check box to disable all trace acquisition conditions. This temporarily disables trace acquisition without deleting the conditions. With **Free Trace** checked all bus cycles are captured, excluding those specified in the **Suppress** section, **Stop** panel, and **Delayed Stop** panel.

The **Trace Events** section of the **General** panel allows you to define events, and event sequences, to be used to initiate trace acquisition.

The **Event** drop-down list box shows all the currently-defined events.

To add a new event click **Add…,** and enter the details of the event in the **Breakpoint/Event Properties** dialog box. For more information about the options available see section 5.3, Complex Event System.

To edit an event select it in the **Event** list and click **Edit….**

To define a sequence of events click **Sequence….** This option is only available if one or more events have been defined with **Enable Sequencing** selected.

To delete an event select it in the **Event** list and click **Delete**, or click **Del All** to delete all the trace events.

5.6.2 Stop

Allows you to stop trace acquisition on the occurrence of a specified event.

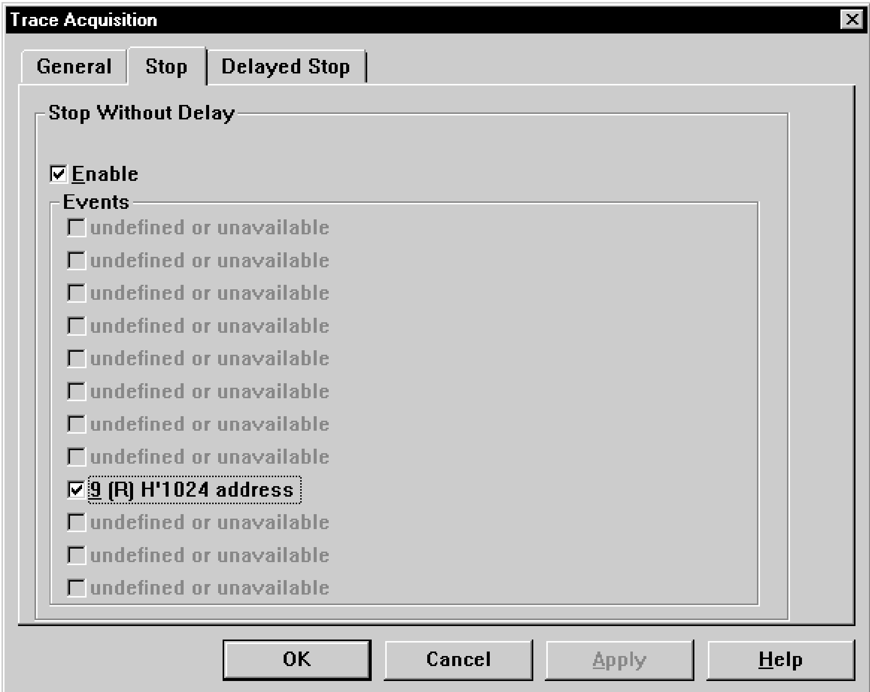


Figure 5.19 Stop Panel

5.6.3 Delayed Stop

Allows you to specify that trace acquisition should continue for a specified number of cycles after a specified event.

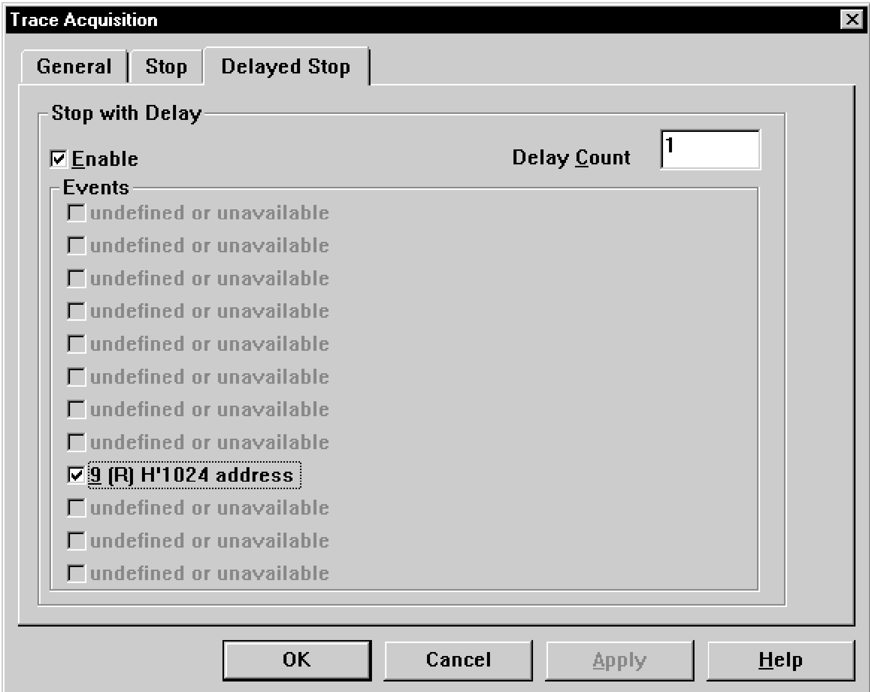


Figure 5.20 Delayed Stop Panel

5.7 Command Line

Figure 5.21 Command Line Window

The **Command Line** window allows you to execute commands to automate debugging. To display the **Command Line** window choose **Command Line** from the **View** menu.

For details of the additional MCU-specific command line functions refer to section 6, Command Line Functions.

Section 6 Command Line Functions

This section gives details of the additional MCU-specific command line functions.

For other general command line functions, refer to Hitachi Debugging Interface User's Manual (HDI manual). Table 6.1 shows the correspondence between the command line functions and the descriptions in the HDI manual and this manual.

Table 6.1 Correspondence Between Command Line Functions and Descriptions in Manuals

Command Name	Abbrevia-tion	HDI Manual	This Manual	Description
!	—	O	—	Comments
ACCESS	AC	O	—	Sets operation for invalid access
ANALYSIS_RANGE	AR	O	—	Sets or displays the performance analysis range
ANALYSIS_RANGE_DELETE	AD	O	—	Cancels the performance analysis range
ANALYSIS	AN	O	—	Validates or invalidates the performance analysis range
ASSEMBLE	AS	O	—	Assembles a program
ASSERT	—	O	—	Checks conditions
BREAKPOINT / EVENT	BP, EN	—	6.1	Sets a breakpoint or an event
BREAKPOINT_CLEAR, EVENT_CLEAR	BC, EC	—	6.2	Clears a breakpoint or an event
BREAKPOINT_DISPLAY, EVENT_DISPLAY	BD, ED	—	6.3	Displays a breakpoint or an event
BREAKPOINT_ENABLE, EVENT_ENABLE	BE, EE	—	6.4	Enables or disables a breakpoint or an event
BREAKPOINT_SEQUENCE, EVENT_SEQUENCE	BS, ES	—	6.5	Defines or clears a breakpoint or event sequence

Notes: 1. O : Described
— : Not described
2. The numbers in the table show the reference section numbers.

Table 6.1 Correspondence Between Command Line Functions and Descriptions in Manuals (cont)

Command Name	Abbrevia-tion	HDI Manual	This Manual	Description
CLOCK	CK	—	6.6	Sets the CPU clock rate in the E6000 emulator
DEVICE_TYPE	DE	—	6.7	Selects the target device in the E6000 emulator
DISASSEMBLE	DA	O	—	Disassembles and displays a program
ERASE	ER	O	—	Clears the contents of the command window
EVALUATE	EV	O	—	Evaluates an expression
FILE_LOAD	FL	O	—	Loads an object program file
FILE_SAVE	FS	O	—	Saves memory contents in a file
FILE_VERIFY	FV	O	—	Verifies memory contents against file contents
GO	GO	O	—	Executes a user program
GO_RESET	GR	O	—	Executes a user program from the reset vector
GO_TILL	GT	O	—	Executes a user program until a temporary breakpoint
HALT	HA	O	—	Stops user program execution
HELP	HE	O	—	Displays the help message for the command line or the command
INITIALISE	IN	O	—	Initializes the platform
INTERRUPTS	IR	O	—	Enables or disables the interrupt processing of the platform (not supported in the E6000 emulator)
LOG	LO	O	—	Manipulates the logging file

Notes: 1. O : Described
— : Not described
2. The numbers in the table show the reference section numbers.

Table 6.1 Correspondence Between Command Line Functions and Descriptions in Manuals (cont)

Command Name	Abbrevia-tion	HDI Manual	This Manual	Description
MAP_DISPLAY	MA	O	—	Displays the memory map information
MAP_SET	MS	—	6.8	Sets memory mapping
MEMORY_DISPLAY	MD	O	—	Displays memory contents
MEMORY_EDIT	ME	O	—	Modifies memory contents
MEMORY_FILL	MF	O	—	Fills the memory with the specified data
MEMROY_MOVE	MV	O	—	Moves a memory block
MEMORY_TEST	MT	O	—	Tests a memory block
MODE	MO	—	6.9	Sets or displays the CPU mode
QUIT	QU	O	—	Terminates the HDI
RADIX	RA	O	—	Sets a radix for input value
REFRESH	RF	—	6.17	Updates memory related windows
REGISTER_DISPLAY	RD	O	—	Displays the CPU register values
REGISTER_SET	RS	O	—	Sets the CPU register values
RESET	RE	O	—	Resets the CPU
SLEEP	—	O	—	Delays command execution
STEP	ST	O	—	Performs single-step execution in instruction unit or source line unit
STEP_OVER	SO	O	—	Performs step-over execution
STEP_RATE	SR	O	—	Executes multiple steps
STEP_OUT	SP	O	—	Performs single-step execution until the end of the function that includes the current PC address

Notes: 1. O : Described
— : Not described
2. The numbers in the table show the reference section numbers.

Table 6.1 Correspondence Between Command Line Functions and Descriptions in Manuals (cont)

Command Name	Abbreviation	HDI Manual	This Manual	Description
SUBMIT	SU	O	—	Executes an emulator command file
SYMBOL_ADD	SA	O	—	Adds a symbol
SYMBOL_CLEAR	SC	O	—	Deletes a symbol
SYMBOL_LOAD	SL	O	—	Loads a symbol information file
SYMBOL_SAVE	SS	O	—	Saves a symbol information file
SYMBOL_VIEW	SV	O	—	Displays a symbol
TEST_EMULATOR	TE	—	6.10	Tests the E6000 emulator hardware
TIMER	TI	—	6.11	Sets or displays the timer resolution for execution time measurement
TRACE	TR	O	—	Displays trace data
TRACE_ACQUISITION	TA	—	6.12	Sets or displays trace acquisition information
TRACE_COMPARE	TC	—	6.13	Compares trace data
TRACE_SAVE	TV	—	6.14	Saves trace data
TRACE_SEARCH	TS	—	6.15	Searches for trace data
USER_SIGNALS	US	—	6.16	Enables or disables user signals

Notes: 1. O : Described
— : Not described
2. The numbers in the table show the reference section numbers.

6.1 BREAKPOINT / EVENT

Abbreviation: BP, EN

Sets a breakpoint. This command has several formats to allow different types of breakpoints to be set.

There are three different types available. These are:

- Program breakpoints,
- Access breakpoints,
- Range breakpoints.

6.1.1 Program Breakpoints

Syntax : bp program *address*

: bp p *address*

This will set a program breakpoint at the address specified.

6.1.2 Access Breakpoints

Syntax : bp access *address* [*options*]

: bp a *address* [*options*]

Options are:

```
<options>    = [<dataopts>] [read|write] [<signalopts>] [<busopts>]
[<areaopts>] [<actionopts>] [count <countval>]
[delay <delayval>] [channel <channelno>]

<dataopts>   = data <data> [mask <mask>] [byte|word]

<signalopts> = signal ((1|2|3|4) (high|low))+

<busopts>    = bus (cpu | cpupre | sadata | sapre | cpumdata |
cpumpre)+

<areaopts>   = area (io | iram | irom | lcdram)+

<actionopts> = action (trace | none | break|(timer (start|stop)))+

<channelno>  = 1..12
```

An access breakpoint causes a break if the MCU accesses the specified address in the specified way.

6.1.3 Range Breakpoints

Syntax: `bp range [outside] <address low> <address hi> [<options>]`

`<options>` is the same as specified in the access breakpoints.

This command will set a breakpoint that will trigger either within the addresses specified, or outside, when the MCU accesses within or outside the specified address range.

6.1.4 Options

- **`data <data> [mask <mask>] [byte | word]`**
  This allows a data comparison to be specified. When bits are masked, the bit data corresponding to the bits that are masked to 0 are not compared.
  Example: `data h'20 mask h'fff0 word`. This will cause the event to occur only if the higher 12 bits of the data bus are set to h'002.
  The default is to not compare the data.

- **`signal ((1 | 2 | 3 | 4) (high | low)) +`**
  With this option the event will only occur if the external probe are in the specified state.
  Example: `signal 1 high 3 low`. This will cause the event to occur only if the signal 1 is high and signal 3 is low. (The value of the other signals is not checked).
  The default is to ignore all signals.

- **`bus (cpu | cpupre | sadata | sapre| cpumdata | cpumpre) +`**
  The event only occurs if the MCU bus is in one of the specified states.

Table 6.2 MCU Bus Status

Events	MCU Status
cpu	CPU data access cycle in active mode
cpupre	CPU instruction prefetch cycle in active mode
sadata	CPU data access cycle in subactive mode
sapre	CPU instruction prefetch access cycle in subactive mode
cpumdata	CPU data access cycle in medium-speed active mode
cpumpre	CPU instruction prefetch access cycle in medium-speed active mode

Example: `bus cpu cpupre`. This will cause the event to occur only if the bus state is prefetch or data access.

The default is to ignore the bus cycle type.

- **`area (io | iram | irom | lcdram) +`**

  Similarly to '`bus...` ' this option causes the event to occur only if the specified areas are being accessed.
  Example: `area irom iram`. This will cause the event to occur only if internal ROM or RAM is being accessed. `lcdram` indicates MCU LCD RAM area access.

  The default is any area.

- **`action (trace | none | break | (timer (start | stop)))+`**

  Defines the action to occur when the event is detected.

  The default action is to break. The other options are to start and stop the event timer that measures the execution time between events. (There is only one timer.)

- **`count <countval>`**

  Sets an event pass count in bus cycles (decimal).

- **`delay <delayval>`**

  Specifies delay cycles in bus cycles for the period after an event has been occurred until operation begins.

- **`channel 1..12`**

  Sets the event detector system channel number to be defined. This is useful if you are setting up a sequence of events since the sequencing is set up by referencing the channel numbers. (See section 6.5, EVENT_SEQUENCE) Channels 1 to 8 are event detectors, 9 to 12 are range detectors.

Examples:

`en access 100`	Sets an access breakpoint at address 100.
`bp p 110`	Sets a program breakpoint at address 110.
`en access 100 data 55 byte`	Sets an access breakpoint at address 100 and data 55 access.
`bp range 12 45`	Sets a range breakpoint from address 12 to 45.
`bp range outside 60 89`	Sets a range breakpoint that will break if address outside 60 and 89 are accessed.
`bp a 200 read`	Sets a access breakpoint to the read cycle of address 200.
`bp a 500 write`	Sets a access breakpoint to the write cycle of address 500.
`bp a 100 read channel 8`	Sets a read access breakpoint at address 100 by channel 8. When the channel 8 condition is satisfied, a trigger signal is output from the external probe.

6.2 BREAKPOINT_CLEAR / EVENT_CLEAR

Abbreviation: BC, EC

This command deletes a breakpoint that has been previously set by the user.

Table 6.3 BREAKPOINT_CLEAR/EVENT_CLEAR Parameters

Keyword	Breakpoint Type
program <address>	Clears a specified program breakpoint
access <address> <options>	Clears a specified access breakpoint
range <address> <options>	Clears a specified range breakpoint
all	Removes all breakpoints
all trace	Removes all trace events
channel 1..12	Removes event by the specified channel number

The <options> are as specified in BREAKPOINT/EVENT command. Only the minimum set of options needed to uniquely identify the event need to be specified.

Examples

`bc p 256`	Clears a program breakpoint at address 256.
`event_clear chan 5`	Removes event by using channel number.
`bc all`	Clears all breakpoints.

6.3 BREAKPOINT_DISPLAY / EVENT_DISPLAY

Abbreviation: BD, ED

Displays enable/disable for the currently set breakpoints. “trace” is displayed for trace events.

Example:

bd	Displays all breakpoints and whether they are enabled or disabled.

6.4 BREAKPOINT_ENABLE / EVENT_ENABLE

Abbreviation: BE, EE

Enables or disables either a single breakpoint, or all the breakpoints.

Table 6.4 BREAKPOINT_ENABLE/EVENT_ENABLE Parameters

Parameter	Keyword	Description
1	true	Enables breakpoint
	false	Disables breakpoint
2	all	All breakpoints
	program <address>	Program breakpoint
	access <address> <options>	An access breakpoint
	range <address1> <address 2>	A range breakpoint
	<options>	
	channel 1..12	Enables or disables an event at the specified channel number

The <options> can be specified as in BREAKPOINT/EVENT command to identify the event more accurately.

Examples:

`be true all`	Enables all breakpoints.
`be false all`	Disables all breakpoints.
`be false p 256`	Disables program breakpoint at address 256.
`be true access 12`	Enables access breakpoint at 12.
`be false chan 1`	Disables event detector channel 1.

6.5 BREAKPOINT_SEQUENCE / EVENT_SEQUENCE

Abbreviation: BS, ES

Syntax:

```
bs <channel> [armed_by [not] <chan1> <chan2> ...]
                         [armed_by off]
                         [reset_by <chan1> <chan2> ...]
                         [reset_by off]
```

Allows you to define events which arm or reset an event.

Examples:

```
bs 1 armed_by 2 3
```

Means events 2 or 3 will arm event 1. The numbers are the channel numbers of the event detectors, 1 to 8 which can be set using the channel option of the event command.

```
bs 2 reset_by 4
```

Event 2 is reset by event 4 when event 4 occurs.

The Off keyword is used to disable arming/resetting of an event by other events, the event then becomes independent.

6.6 CLOCK

Abbreviation: CK

Selects or displays the source or rate of the active system clock (ϕ) and the subclock (ϕw). With no parameters the active clock source and rate are displayed. If the clock source or rate is changed the E6000 emulator system is reset.

In the MCU, the rate of system clock (OSC1 and OSC2) is a half of the input clock rate.

Table 6.5 CLOCK Parameters

Parameter	Keyword	Clock Source (Optional)
1	05	0.5-MHz internal clock
	2	2-MHz internal clock
	8	8-MHz internal clock
	t2	Target /2
2	sub 32k	32.768-kHz internal subclock (ϕw)
	sub 38k	38.4-kHz internal subclock (ϕw)
	sub 307k	307.2-kHz internal subclock (ϕw)
	sub t	target subclock

Note that the user system clock can only be selected if the Vcc is supplied from the user system.

Examples

`ck`	Displays the current emulation clock.
`ck 2 sub 32k`	Selects 2 MHz system clock and 32.768 kHz subclock.

When a break is detected when the subclock (32.768 kHz or 38.4 kHz) is used, the emulator operation and display become slow. Therefore, when evaluating using the subclock, select 307.2 kHz which is the eight times the frequency of 38.4 kHz.

Notes: 1. The target system clock can be selected only when the Vcc is supplied from the user system.

2. When using the target MCU (H8/3802 series), 307.2 kHz cannot be selected as the subclock (ϕw).

6.7 DEVICE_TYPE

Abbreviation: DE

Sets up the device type to emulate, or displays current setting.

Examples:

`de`		Displays device type.
`de`	`h8/3802`	Sets device to H8/3802.

6.8 MAP_SET

Abbreviation: MS

This option will emulation memory mapping.

Syntax:

```
ms <start>  <end>  (internal | internal)   (none | read-only | guarded)
```

Examples:

`ms 8000 F73F internal`	Allocates internal read/write memory from H'8000 to H'F73F.

Note: **`internal`** is used for memory areas on the chip, i.e. internal ROM, RAM, I/O, or reserved area. The attribute of these areas cannot be changed except that the reserved area excluding H'EE00 to H'F73F, H'F760 to H'F77F, and H'FF80 to H'FF8F can be changed to the emulation memory by specifying `Internal`.

6.9 MODE

Abbreviation: MO

Sets and displays the MCU mode.

Table 6.6 MODE Parameter

Parameter	Keyword	Mode Type
1	3	3 (Single chip mode)

In the MCU, mode is fixed to 3.

Examples:

`mode`	Lists the current mode.
`mode 3`	Sets mode to 3, and maps memory again.

6.10 TEST_EMULATOR

Abbreviation: TE

Tests the E6000 emulator hardware, and performs a test of the E6000 emulator memory areas. After running this command, the E6000 emulator system must be re-initialized.

Examples:

`te`	Performs E6000 emulator testing.

6.11 TIMER

Abbreviation: TI

Allows the timer resolution to be displayed and modified. This will set the timer resolution for measuring the execution time and execution time between events.

Table 6.7 TIMER Commands

Command	Description
ti	Displays the timer resolution
ti <timer resolution>	Sets the timer resolution

Timer resolutions are: 20 ns, 125 ns, 250 ns, 500 ns, 1 µs, 2 µs, 4 µs, 8 µs, or 16 µs

Examples:

`ti 20`	Sets the timer resolution to 20 ns.
`ti 250ns`	Sets the timer resolution to 250 ns.
`ti 8`	Sets the timer resolution to 8 µs.
`ti 16us`	Sets the timer resolution to 16 µs.

6.12 TRACE_ACQUISITION

Abbreviation: TA

Sets or displays trace acquisition options.

Syntax:

```
TA [<suppress>] [<freetrace>] [<timestamp>] [<stop>] [<stopdelay>]
[<range>] [<default>]


<suppress>  = suppress dtu (true|false) (cannot be used for the MCU)
<freetrace> = freetrace (true|false)
<timestamp> = timestamp (disable | 125ns | 250ns | 500ns | 1us |
              2us | 4us | 8us | 16us | 100us )
<stop>      = stop ( disable | event <1 to 12>)
<stopdelay> = stopdelay ( disable | event <1 to 12>
              [count <count>] )
<range>     = range <1 to 4> ( disable |
                ptop <startaddr> <stopaddr> [cyclic] |
                range <1 to 12> |
                event <1 to 8> <1 to 8> [cyclic])
<default>   = default
```

Examples:

ta	Displays all trace acquisition options.
ta stop event 1 2	Stops tracing when either event on channel 1 or channel 2 occurs.
ta stopdelay event 1 2 count 100	Stops tracing 100 bus cycles after event on channel 1 or channel 2 occurs.
ta timestamp 500ns	Enables trace timestamping and sets the resolution of the timer stamp to 500 ns.
ta range 2 event 4 5 cyclic	Lists trace range 2 to start trace when event 4 occurs and stop trace when event 5 occurs and then to restart trace when event 4 occurs again.

6.13 TRACE_COMPARE

Abbreviation: TC

Compares a saved trace file (see trace_save) with the current trace data.

```
trace_compare <filename>
```

6.14 TRACE_SAVE

Abbreviation: TV

Saves the trace data to a file in binary format. The saved data can be compared with the trace using the trace_compare command.

```
trace_save <filename>
```

6.15 TRACE_SEARCH

Abbreviation: TS

Search Trace results. This command will let the user search in the same way as the **trace find** dialog box.

Syntax:

```
TS [<address>] [<dataopts>] [<signalopts>] [<busopts>] [<areaopts>]
[<directionopts>] [<timestampopts>] [<fromopts>]


<address>       = address <address> [to <address>]
<dataopts>      = data <data> [mask <mask>] [byte|word]
<signalopts>    = signal <sig><sig><sig><sig>
       <sig>    = (1|0|x) 1 = high, 0 = low, x = don't care
<busopts>       = bus (cpu | cpupre | sadata | sapre | cpumdata |
                  cpumpre)+
<areaopts>      = area (io | iram | irom | lcdram)+
<directionopts> = dir (read | write | either)
<timestampopts> = time <start> [ to <stop>]
   <start> and <stop> should be in format 0s:0000.000
<fromopt> =from <record>
```

Examples:

`ts address 104 data 55aa w`	Searches trace data for the cycles that accessed data 55aa at address 104 in word units.
`ts area irom`	Searches trace data for the cycles that accessed the ROM area.

6.16 USER_SIGNALS

Abbreviation: US

Allows the user signals (Reset) to be enabled or disabled. With no parameters this command will display the state of the enabled/disabled flags for Reset.

Table 6.8 USER_SIGNALS Commands

Command	Description
us	Displays user signal status.
us enable reset	Enables the signals specified.
us disable reset	Disables the signals specified.

6.17 REFRESH

Abbreviation: RF

Updates the memory related windows.

Section 7 Diagnostic Test Procedure

This section describes the diagnostic test procedure using the E6000 test program.

7.1 System Set-Up for Test Program Execution

To execute the test program, use the following hardware; do not connect the user system interface cable and user system.

- E6000 (HS3800EPI60H)
- Host computer
- The E6000 PC interface board which will be one of the following boards or card:

  Select one interface board from the following depending on the PC interface specifications.

  ISA bus interface board (HS6000EII01H)

  PCI bus interface board (HS6000EIC01H or HS6000EIC02H)

  PCMCIA interface card (HS6000EIP01H)

1. Install the E6000 PC interface board in the host computer and connect the supplied PC interface cable to the board.
2. Connect the PC interface cable to the E6000.
3. Connect the supplied AC adapter to the E6000.
4. Initiate the host computer to make it enter DOS prompt command input wait state.
5. Turn on the E6000 switch.

7.2 Diagnostic Test Procedure Using the Test Program

Insert the test program floppy disk (HS3800EVI60SF supplied with the E6000) into the host computer, move the current directory to A:, and enter one of the following commands according to the PC interface board used to initiate the test program:

1. ISA bus interface board (HS6000EII01H)
   >A:TM3800 –ISA (RET)

2. PCI bus interface board (HS6000EIC01H or HS6000EIC02H)
   >A:TM3800 –PCI (RET)

3. PCMCIA interface card (HS6000EIP01H)
   >A:TM3800 –PCCD (RET)

Be sure to initiate the test program from directory A. Do not initiate it from a directory other than A:, such as C:¥>A: TM3800 –ISA (RET). If the test program is initiated when the current directory is not A:, the test program will not operate correctly.

It will take about 6 minutes to execute the test program when the host computer using Windows® 95 runs at 166 MHz and the PCMCIA interface card is used. The following messages are displayed during the test. Tests are from no.1 to no.14.

Message	Description
`E6000 H8/3800 EMULATION BOARD Tests Vx.x` `Hitachi Ltd (2000)`	Test program start message. Vx.x shows the version number.
`SIMM module fitted? ( 1. None  2. 1MB  3. 4MB ) : 1`	Enter 1 because the SIMM memory module is not installed in this example.
`Searching for interface card ..............................OK`	Shows that the PC interface board is correctly installed in the host computer, and displays the address when the ISA bus interface is installed. The displayed address depends on the settings. When the PCI interface board or PCMCIA interface card is installed, the address is not displayed.
`Checking emulator is connected ............................OK`	Shows that the E6000 is correctly connected to the host computer.
`Emulator Board Information:` `Main Board ID        H'1`	Shows the ID number of the lower board of the E6000 (always 1).
`Emulation Board ID  H'15`	Shows the ID number of the upper board of the E6000 (always 15).
`SIMM                No SIMM module inserted`	Shows whether the SIMM memory board is installed.

`01) Testing Main Board Register :`
`IDR0 Register.........................................OK`
`PAGE Register.........................................OK`
`CES G/A Register .....................................OK`
`IDR1 Register.........................................OK`
	Shows the check results for the registers in the E6000 (normal completion).

RENESAS

```
02) Testing Dual-Port RAM :

      Decode Test .........................................OK
      Marching Test .......................................OK
```

Shows the results of decoding test and step test for the dual-port RAM in the E6000 (normal completion).

```
03) Testing Firmware RAM :

      Decode Test.   page range H'700 - H'71f ..............OK
```

Shows the results of decoding test for the firmware RAM in the E6000 (normal completion).

```
      Marching Test. page range H'700 - H'71f ..............OK
```

Shows the results of step test for the firmware RAM in the E6000 (normal completion).

```
04) Testing Trace RAM :
      Decode Test.   page range H'000 - H'04f ..............OK
```

Shows the results of decoding test for the trace RAM in the E6000 (normal completion).

```
      Marching Test. page range H'000 - H'04f ..............OK
```

Shows the results of step test for the trace RAM in the E6000 (normal completion).

```
05) Testing Mapping RAM :
      Decode Test.   page range H'200 - H'27f ..............OK
```

Shows the results of decoding test for the mapping RAM in the E6000 (normal completion).

```
      Marching Test. page range H'200 - H'27f ..............OK
```

Shows the results of step test for the mapping RAM in the E6000 (normal completion).

```
06) Testing Internal ROM and RAM :
     Setting up, please wait..
     Decode Test   [0x0000 - 0xff7f].......................OK
     Marching Test [0x0000 - 0xff7f].......................OK
```

Shows the results of decoding test and step test for internal ROM and RAM in the E6000 (normal completion).

```
07) Testing STEP Operation :
      Setting up, please wait..
      Step Operation .......................................OK
```

Shows the check results for the step execution controlling circuits in the E6000 (normal completion).

```
08) Testing Key Break :
      Setting up, please wait..
      Key Break ............................................OK
```

Shows the check results for the forced break controlling circuits in the E6000 (normal completion).

```
09) Testing Emulation RAM Hardware Break :
      Setting up, please wait..
      GRD Break ............................................OK
      WPT Break ............................................OK
```

Shows the check results for the illegal access break controlling circuits in the E6000 (normal completion).

```
10) Testing Internal ROM Write-Protect :
      Setting up, please wait..
      Write-Protect ........................................OK
```

Shows the check results for the internal ROM write-protection controlling circuits in the E6000 (normal completion).

```
11) Testing Hardware Break :
      Setting up, please wait..
      A)Break Point Initialised ............................OK
      B)Event Detectors CES channel 1-12 ...................OK
      C)Test Sequencing 1 ..................................OK
      D)Check Range Break ..................................OK
      E)Range Break Test for Data ..........................OK
```

Shows the check results for the hardware break control circuits in the E6000 (normal completion).

```
12) Testing Emulation RAM Trace :
      Setting up, please wait..
      A)Free Trace Test ....................................OK
      B)Range Trace Test ...................................OK
      C)Point to Point Trace Test ..........................OK
      D)Start and Stop Event Trace Test ....................OK
      Setting up, please wait..
      E)Time STAMP Trace Test
        Time STAMP Trace Test 1 ............................OK
        Time STAMP Trace Test 2 ............................OK
        Time STAMP Trace Test 3 ............................OK
```

Shows the check results for the trace controlling circuits in the E6000 (normal completion).

`13) Testing Runtime counter :`
`Setting up, please wait..`
`Testing Internal Clock =  8.00MHz ....................OK`
`Testing Internal Clock =  2.00MHz ....................OK`
`Testing Internal Clock =  0.5MHz .....................OK`
`Testing Internal SubClock =  32.768kHz ...............OK`
	Shows the check results for the run-time counter in the E6000 (normal completion).

`14) Testing Emulation Monitor :`
`Setting up, please wait..`
`A)A15-A0 (MONIT1OE:D7-D0) TEST........................OK`
`B)ST2 to ST0 (MONIT2E:D6-D4, MONIT0E:D2-D0) ..........OK`
`C)BRKACK (MONIT0E:D4) TEST............................OK`
`D)CNN  (MONIT2E:D1) TEST .............................OK`
`E)IF (MONIT2E:D7) TEST ...............................OK`
`F)WINDOW (MONIT2O:D1) TEST ...........................OK`
	Shows the check results for the emulation monitor controlling circuits in the E6000 (normal completion).

`Tests run for xH:xM:xS`	Shows the check time.
`0 total errors`	Total number of errors.
`Tests passed, emulator functioning correctly`	Shows that the E6000 is correctly operating.

When detecting an error, the test program displays ERROR and stops execution. In this case, the emulator hardware may be malfunctioning. Inform a Hitachi sales agency of the test results in detail.

H8/3802 Series E6000 Emulator User's Manual

Publication Date: 1st Edition, September 2000
Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.